Hacker

Σεπτέμβριος 2015 # Τιμή τεύχους: 3,99 €
τεύχος 048 - Encrypted Dreams Edition

OpenVPN Access Server στο cloud [*]

Προχωρημένα χαρακτηριστικά της C#

Κρυπτογράφηση της κίνησης *όλου* του LAN

Προγραμματιστικές τεχνικές για τη σύνθεση σήματος VGA

Κόλπα για την επιτάχυνση της GPU, του δικού μας υπολογιστή

[*] ...χωρίς πόνο ή βάσανα και για όλους!

all rights reversed

HACKER


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Πέτρος Κυλαδίτης, Νίκος Μουρατίδης, Θωμάς Σερμπίνης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας, @arkoudos
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στην Καλαμαριά Θεσσαλονίκης. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/pdf12order και από την ίδια σελίδα γίνονται οι παραγγελίες συνδρομών.



deltaHacker Field Bag
€25.50
MORE COLORS AVAILABLE
deltaHacker Mugs
€12.50
deltaHacker Thermos® Can Cooler
€17.50
QUESTIONS? | ORDER BY PHONE? | CALL 1-877-809-1659
Where's My Stuff?
Track your recent orders.
View or change your order.
Shipping & Returns
See our shipping rates.
Make a return or exchange.
Need Help?
Satisfaction Guaranteed.
Visit our Customer Service.
is powered by CafePress Inc. Copyright © 1999-2015 CafePress.com. All rights reserved.
olicy | Trademark & Copyright Information
urrency rates are updated daily and may fluctuate.
Δώρα με στιλ, που έχουν *αληθινή* αξία.
Όλα στο μαγαζάκι του deltaHacker,
κάτω από το Cafepress.
Περάστε μια βόλτα από το
http://www.cafepress.co.uk/deltahacker
–σας περιμένουμε!

Ελάχιστο Στοιχείο

Έψαχνα κάτι τις προάλλες και, όπως συνήθως συμβαίνει κάθε φορά που κάποιος κάτι ψάχνει τις προάλλες, αυτό που βρήκα δεν ήταν αυτό που αναζητούσα. Μάλιστα η ανάμνηση του συγκεκριμένου "κάτι" ήταν βαθιά θαμμένη στο πίσω μέρος του μυαλού μου.

του Χρήστου Βαρελά

Αγαπητές κι αγαπητοί,

Θα γνωρίζετε, φαντάζομαι, ότι εδώ κι αρκετά χρονάκια πολλοί συνάνθρωποί μας στο πίσω μέρος του μυαλού τους έχουν, εξ ανάγκης, μάλλον πεζές σκέψεις -- και δεν έχει σημασία αν σημαντικό ποσοστό αυτών είναι δεύτερες. Αλλά μη νομίζετε: Εκτός από τις μάλλον πεζές ή/και δεύτερες σκέψεις, εκεί, στο πίσω μέρος του μυαλού, κατοικούν και κάποιες σκέψεις --ή καλύτερα αναμνήσεις-- οι οποίες παραμένουν καλά κρυμμένες και δεν αποκτούν καν μια ευκαιρία για να βγουν μπροστά.

Εκτός δηλαδή κι αν ψάχνεις κάτι, κατά προτίμηση τις προάλλες.

Έτσι λοιπόν και με τη δική μου περίπτωση. Για άλλο ξεκίνησα να ψάχνω, άλλο τελικά βρήκα. Το τι έψαχνα δεν έχει και πολλή σημασία. Για να σας πω την αλήθεια, αυτή τη στιγμή δεν καλοθυμάμαι καν τι ήταν. Αυτό που βρήκα, κι από τις προάλλες δεν το 'χω ξεχάσει, ήταν ένα βιβλίο της Θεωρίας Αριθμών.

Το βιβλίο προοριζόταν για τους φοιτητές του Τμήματος Μαθηματικών, της Σχολής Θετικών Επιστημών, του Αριστοτελείου Πανεπιστημίου της Θεσσαλονίκης. Βεβαίως, το αντίστοιχο μάθημα γινόταν και γίνεται και σε άλλα τμήματα μαθηματικών, σε άλλες σχολές και πανεπιστήμια. Στο Αριστοτέλειο, πάντως, το μακρινό 1991 ήταν ένα από τα υποχρεωτικά μαθήματα του πρώτου εξαμήνου.

Δεν μπορώ να ξεχάσω ότι από ένα σημείο και μετά είχα δυσκολευτεί με τη Θεωρία Αριθμών, κυρίως επειδή δεν στρωνόμουν στη μελέτη -- κι αυτό ήταν κάτι που μου συνέβαινε με σχεδόν όλα τα μαθήματα. Είχα ωστόσο προλάβει να την αγαπήσω. Διαβάζοντας ξανά τον πρόλογο της έβδοσης έκδοσης του βιβλίου του καθηγητή Κ. Λάκκη, θυμήθηκα ότι η Θεωρία Αριθμών συχνά χαρακτηρίζεται ως η "βασίλισσα των Μαθηματικών".

Δικαίως, κατά την ταπεινή μου γνώμη. Αν και τα αντικείμενα ή καλύτερα οι υπό μελέτη έννοιες της Θεωρίας Αριθμών είναι κατά βάση απλές, οι αυστηρές αποδείξεις προτάσεων και θεωρημάτων δεν είναι προφανείς ή εύκολες. Πάντα, ωστόσο, είναι κομψές και τελικά άκρως γοητευτικές. Αυτό διαφαίνεται πριν καν περάσουμε στο βασικό αντικείμενο του βιβλίου. Ως παράδειγμα, αντιγράφω εδώ ένα θεώρημα από τη σελίδα 3:

Θεώρημα 1.2. Κάθε μη κενό υποσύνολο του συνόλου των φυσικών αριθμών έχει ελάχιστο στοιχείο.

Στο σημείο αυτό κάθε φυσιολογικός άνθρωπος θα κουνήσει το κεφάλι του ελαφρώς απελπισμένος, και μάλλον θα σκεφτεί ότι δεν υπάρχει καμία σωτηρία για τους μαθηματικούς. "Χε-λό-όου! Προφανώς και κάθε μη κενό υποσύνολο του συνόλου των φυσικών αριθμών έχει ελάχιστο στοιχείο. Θέλει μωρέ απόδειξη αυτό;! Τι είναι, δηλαδή, τα μη κενά υποσύνολα του συνόλου των φυσικών αριθμών; Βαρέλια χωρίς πάτο, να τους ρίχνουμε στοιχεία κι αυτά να φεύγουν από κάτω;" Κάτι παρόμοιο είχα σκεφτεί κι εγώ, όταν για πρώτη φορά διάβασα το Θεώρημα 1.2. Κι όμως, αυτή η διαισθητικά προφανέστατη πρόταση θέλει την απόδειξή της -- και την έχει. Την αντιγράφω εδώ, παρεμβάλλοντας ελαφρύ σχολιασμό.

Έστω ότι υπάρχει ένα μη κενό υποσύνολο M του συνόλου των φυσικών αριθμών, το οποίο δεν έχει ελάχιστο στοιχείο.

Ξεκινάμε ύπουλα, με το τέχνασμα της εις άτοπον απαγωγής. Κάνουμε, δηλαδή, μια αρχική υπόθεση κι ακολουθώντας έγκυρα βήματα καταλήγουμε σε κάτι που δεν ισχύει. Και γιατί αυτό το κάτι, στο οποίο καταλήγουμε ακολουθώντας έγκυρα βήματα, δεν ισχύει; Μα, γιατί η αρχική μας υπόθεση ήταν λανθασμένη. Εν προκειμένω, το λάθος μας θα είναι ότι υποθέσαμε πως υπάρχει μη κενό υποσύνολο του συνόλου των φυσικών αριθμών χωρίς ελάχιστο στοιχείο. Για να δούμε...

Θ' αποδείξουμε τότε ότι ισχύει:

(1.2.1) για κάθε m του M ισχύει m ≥ n, για κάθε φυσικό αριθμό n

Πράγματι: Η πρόταση αυτή ισχύει για n=0 διότι κάθε m του M είναι μηδέν ή μεγαλύτερο του μηδενός. Υποθέτουμε τώρα ότι η πρόταση (1.2.1) ισχύει για τον φυσικό αριθμό n, δηλαδή υποθέτουμε ότι για κάθε m του M ισχύει m ≥ n.

Έχουμε, φίλες και φίλοι, καταφύγει στη μέθοδο της επαγωγής. Δηλαδή: Ξεκινάμε αποδεικνύοντας ότι η αρχική πρόταση ισχύει για n=0, μετά υποθέτουμε ότι ισχύει για n>0, οπότε αν αποδείξουμε ότι ισχύει και για n+1 τότε συμπεραίνουμε ότι η πρόταση ισχύει *για κάθε* φυσικό αριθμό n. Συνεχίζουμε:

Από τη σχέση αυτή προκύπτει ότι ο n δεν ανήκει στο σύνολο M, διότι διαφορετικά θα ήταν ελάχιστο στοιχείο του M.

Είπαμε: Υποθέσαμε ότι το M *δεν* έχει ελάχιστο στοιχείο!

> Άρα ισχύει m > n για κάθε m του M και συνεπώς m ≥ n+1. Άρα η πρόταση **(1.2.1)** ισχύει και για τον φυσικό αριθμό n+1 και σύμφωνα με τη μέθοδο της μαθηματικής επαγωγής ισχύει η **(1.2.1)** για όλους τους φυσικούς αριθμούς.

Και τώρα το τελειωτικό χτύπημα:

> Η πρόταση όμως αυτή οδηγεί στο εξής άτοπο. Για κάθε m του M ισχύει m ≥ m+1.

Το συμπέρασμα; Ιδού:

> Άρα δεν υπάρχει μη κενό υποσύνολο του συνόλου των φυσικών αριθμών, το οποίο δεν έχει ελάχιστο στοιχείο.

Εκπληκτικό; Ξεκινήσαμε από κάτι προφανές, το αμφισβητήσαμε και κατόπιν αποδείξαμε, αυστηρά και πέρα από κάθε αμφιβολία, την αλήθεια του αρχικού "κάτι".

Θα αναρωτιέστε, ίσως, τι αξία έχουν τέτοιες νοητικές ασκήσεις. Εντάξει, τα Μαθηματικά είναι πράγματι χρήσιμα για τις άλλες επιστήμες (όπως επίσης και για τα ίδια τα Μαθηματικά), όμως τι νόημα έχει να κοπανάμε το κεφάλι μας στον τοίχο, μόνο και μόνο για ν' αποδείξουμε ότι *και* αύριο ο ήλιος θα βγει από την ανατολή;

Κάπου εδώ, οι σεβαστοί κύριοι καθηγητές --κι όχι μόνο οι μαθηματικοί-- θα είχαν πολλά να πουν. Ο γράφων βέβαια δεν είναι ούτε καθηγητής (ούτε και σεβαστός), θυμάται όμως κάτι που συχνά άκουγε από άλλους σεβαστούς καθηγητές: "Μάθετε Μαθηματικά, για να μάθετε να σκέφτεστε και να μη σας πιάνουν κορόιδο."

Καλά τα έλεγαν, οι σεβαστοί κύριοι καθηγητές. Μόνο που στην πράξη κάτι τέτοιο είναι εύκολο να το λες έτσι, γενικά, αλλά τρομακτικά δύσκολο να το υιοθετείς στην καθημερινότητά σου. Εγώ, π.χ., μετά από ζωηρές συζητήσεις και διαλόγους, πάνω σε σοβαρά θέματα, συχνά συνειδητοποιώ ότι ήμουν εντελώς ό,τι να 'ναι, αφού δεν είχα ειρμό κι έκανα το γελοίο λάθος να ρίξω τη μπάλα στην εξέδρα. Η μαθηματική σκέψη υποτίθεται ότι σε προστατεύει από τέτοια λογικά σφάλματα, αφού οι μαθηματικοί συλλογισμοί έχουν, αν μη τι άλλο, αρχή, μέση και τέλος.

Άντε όμως να διατηρήσεις τον ειρμό σου στην Αληθινή Ζωή (TM), όταν, π.χ., έχεις γνωστούς που στο πρόσφατο δημοψήφισμα ήταν με το Όχι, μετά όμως χάρηκαν με τη συμφωνία που ακολούθησε και στο καπάκι ψήφισαν και πάλι Αλέξη.

Αλλά τι τα θέλετε, φίλες και φίλοι; Τα Μαθηματικά είναι λίγα για την Αληθινή Ζωή (TM), όπου τα ελάχιστα στοιχεία ουκ ολίγες φορές τοποθετούνται σε μέγιστες θέσεις.

Εντάξει, αυτό το τελευταίο ήταν φτηνό — και σίγουρα καθόλου μαθηματικό.



ονέιλ μέιλ

Αγαπητοί μου,

Είναι μέρες που το σκέφτομαι πως πρέπει να βγάλω τη γραφομηχανή από το κουτί και να κάτσω επιτέλους να ετοιμάσω την επιστολή για το τεύχος 48, αλλά δεν τα καταφέρνω.

Δεν ξέρω πώς γίνεται. Τη βγάζω από το κουτί, τη στήνω να γράψω, και μετά όλο κάτι συμβαίνει και τελικά τίποτα.

Συνωμοσία ένα πράγμα.

Τη μία μου έχει τελειώσει το χαρτί οπότε δεν το επιχειρώ καν. Την άλλη με διακόπτουν για να με στείλουν για ψώνια, στην Εβγα του Κυρίου Κώστα ας πούμε, και μετά εγώ κολλάω εκεί, καθυστερώ να επιστρέψω, και όταν τελικά επιστρέφω είναι ώρα για φαγητό, οπότε το γράψιμο αναβάλλεται.

Κάποιες άλλες φορές με το που πάω να ξεκινήσω το γράψιμο, εμφανίζεται κάποιος από το πουθενά --η εξαδέλφη, η μητέρα της εξαδέλφης, η Κυρία Καίτη, δεν έχει σημασία-- και ανοίγει την τηλεόραση.

Ξέρετε έχω ένα πρόβλημα με την τηλεόραση, πρέπει να το συζητήσω με το γιατρό μου.

Με το που ανοίγει τηλεόραση, κολλάω, και μένω να κοιτάω σαν τον χάνο.

Αυτές τις μέρες, ας πούμε, τους έχει πιάσει όλους κάτι με τις ειδήσεις. Από τις εκλογές και μετά. Βάζουνε συνεχώς να δούνε ειδήσεις. Κι εγώ κολλάω.

Χτες δείχνανε κάτι παράθυρα που είχανε βγει όλοι και μιλάγανε για τον ΦΠΑ στα ιδιωτικά και τα φροντιστήρια. Που η κυβέρνηση, όλων των Ελλήνων παρακαλώ, θέλει να τον κάνει κλιμακωτό αλλά έρχονται οι ξένοι, και μας λένε να το ξεχάσουμε.

Με πιάνει, που λέτε, απόγνωση. Παρακολουθώ τις συζητήσεις, και όλοι λένε το μακρύ και το κοντό τους, αλλά κανείς δεν κάνει τον κόπο να ρωτήσει το εξής απλό: Τι ζόρι τραβάνε ρε παιδιά οι ξένοι με τον ΦΠΑ στα ιδιωτικά; Τι έχουν πάθει και μας ταλαιπωρούν έτσι;

Μπα... Χαμπάρι. Σαν την Εβγα του Κυρίου Κώστα ένα πράμα. Όλοι συμφωνούν στο εξής: οι ξένοι είναι κακοί και θέλουν το κακό μας. Ο Σόιμπλε, η Μέρκελ, και λοιποί. Αυτοί φταίνε. Από κει και πέρα, οι επί μέρους διαφωνίες έγκεινται στο τι πρέπει να κάνουμε εμείς για να διαχειριστούμε τόση κακία (των ξένων).

Αν τώρα βγει κανείς να πει κάτι άλλο, κάτι διαφορετικό ρε παιδί μου, θα πέσουν να τον φάνε.

Να βγω, ας πούμε εγώ, και να το πω.

Το λέω: Οι ξένοι δεν έχουν κανένα λόγο να επιμένουν στον ΦΠΑ στα ιδιωτικά. Το μόνο που τους ενδιαφέρει είναι να βγουν τα νούμερα. Σ' αυτό συμφωνήσαμε όλοι. Ότι θα βγουν τα νούμερα. Από εκεί και πέρα το θέμα είναι δικό μας. Αν δεν σου αρέσει ο ΦΠΑς, κόψε από αλλού. Κόψε από την επιδότηση του Σωματείου της ΔΕΗ. Κλείσε τη βιομηχανία ζάχαρης. Βάλε χα-

ράτσι στα μοναστήρια. Αλλά άστο το παραμύθι, ότι σε αναγκάζουν οι ξένοι. Αν δεν θέλεις να σ' αναγκάζουν μη ζητάς άλλα δανεικά, που θα' λεγε και ο Βαρουφάκης.

(Τι θα πει, ποιος είναι ο Βαρουφάκης; Γκουγκλ ιτ!)

Μπα. Τίποτα. Δεν το λέω γιατί θα πέσετε να με φάτε. Αμέσως η συζήτηση θα πάει στους τεχνοκράτες των Βρυξελλών που βλέπουν μόνο νούμερα και χάνουν τον άνθρωπο. Μετά θα πάει στον εκβιασμό των δανειστών που μας κλείσανε τις τράπεζες. Θα συνεχίσει με τη σύγκρουση Βορρά και Νότου. Και αναπόφευκτα θα φτάσει στα νεοφιλελεύθερα σχέδια του παγκοσμιοποιημένου καπιταλισμού.

Για να μην πω για τα λεφτά που μας χρωστάνε οι Γερμανοί.

Τρικυμία εν κρανίω.

Και καλά ο Βαρουφάκης. Αυτός κουκιά τρώει, κουκιά μαρτυράει.

Εμείς οι άλλοι, όμως; Νιονιό δεν έχουμε να το βάλουμε να σκεφτούμε;

Απλή στατιστική. Είναι δυνατόν να φταίνε πάντα οι άλλοι; Μα πάντα; Είναι δυνατόν να μας έχουν τόσο πολύ άχτι; Τι έχουμε, πια, το κοκαλάκι της νυχτερίδας;

Λοιπόν αυτό πάντα με κάνει να θαυμάζω και να απορώ. Το πώς σκέφτεται ο κόσμος. Το πώς διαμορφώνεται η "κοινή γνώμη". Και κυρίως, το πώς αλλάζει γνώμη η "κοινή γνώμη".

Απαραίτητο disclaimer: Γνωρίζω τόσο καλά όσο και ο κάθε κοινωνιολόγος της πολυκατοικίας απέναντι, πως έννοιες όπως "ο κόσμος" ή "η κοινή γνώμη" δεν υφίστανται παρά μόνο ως ασαφείς συλλογικές περιγραφές. Δεν υπάρχει "κοινή γνώμη", όπως και δεν υπάρχει "λαός" ή "έθνος". Πρόκειται για κενά σημαίνοντα με χλωμή στατιστική χρησιμότητα.

Το γνωρίζω, και γράμματα γνωρίζω.

Θα μου επιτρέψετε λοιπόν να μην κολλήσω σε αυτό. ΟΚ; Συνεχίζω λοιπόν.

Πώς αλλάζει γνώμη η κοινή γνώμη; Αναρωτιέμαι.

Να σας πω πώς αλλάζει. Αλλάζει στο έτσι, χαλαρά.

Κάνει ο άλλος δημοψήφισμα, βγαίνει το 60% και βροντοφωνάζει: ΟΧΙ! Δεν πληρώνω, εδώ είναι Βαλκάνια, κ.λπ. κ.λπ.

Αρχίζει ο Alexis τις διαπραγματεύσεις, γράφει και κάποιες υπερωρίες, επιστρέφει περιχαρής.

Αμάρτησα για την πατρίδα μου. Εδώ το καλό μνημόνιο.

Το σκέφτονται οι άλλοι λίγο, βγαίνουν σύσσωμοι στο τουίτερ, αρχίζουν και πάλι να βροντοφωνάζουν: πραξικόπημα, #ThisIsACoup κ.λπ. κ.λπ.

Μετά ξανά μανά εκλογές. Λες, εδώ σε θέλω κάβουρα, κάτσε να δω. Στήνεσαι στην τηλεόραση, βγαίνουν τα αποτελέσματα, βλέπεις. Σύσσωμο και πάλι το εκλογικό σώμα, βροντοφωνάζει: ΝΑΙ, δικέ μου, μέσα. Μνημόνιο. ΕΝΦΙΑ. ΦΠΑ. Μέσα. Τα όλα.

Απ' τη ζωή βγαλμένα όλα αυτά αγαπητοί μου, όπως σας τα λέω έγιναν.

Κοινή γνώμη, σου λέει ο άλλος.

Τι είναι αυτό που μας κάνει και κάποια πράγματα τα πιστεύουμε ο κόσμος να χαλάσει, ενώ κάποια άλλα τα αρνούμαστε ακόμα κι αν τα λέει ο προφήτης ο ίδιος;

Ο Alexis, για παράδειγμα. (Γκουγκλ!)

Ο Alexis στο απυρόβλητο. Λαϊκό παιδί. Τίμιο. Θα μας βγάλει από την τρύπα. Θα σκίσει τα μνημόνια (των άλλων). Θα τους δείξει. Επαναστάτης.

Στον αντίποδα, οι κακοί ξένοι. Οι δανειστές. Οι τοκογλύφοι. Το μεγάλο κεφάλαιο. Οι νεοφιλελεύθεροι. Οι καλικάντζαροι. Αυτοί φταίνε για όλα.



Για όλα λέμε. Για τους τυφλούς της Ζακύνθου. Για την συμπεριφορά των υπαλλήλων στα εκδοτήρια των ΚΤΕΛ. Για τις αυτοκτονίες (που σταμάτησαν). Για τις αυθαιρεσίες των αστυνομικών. Για τις μίζες των εξοπλιστικών. Για τις συντάξεις στη ΔΕΗ. Για τις λιποθυμίες στα σχολεία.

Δέκα κωλοτούμπες να κάνει ο Alexis, εμείς εκεί, ορθοί υπερασπιστές, δίπλα του. Με τις σάρισες.

Δέκα φορές να μας κόψει το χρέος το ΔΝΤ εμείς εκεί, απέναντί του. Βράχοι. Στιβαροί. Αποφασισμένοι.

Πώς συμβαίνει αυτό; Πώς εξηγείται;

Κατά βάση νομίζω πως αυτό που θέλει ο κόσμος είναι μια αίσθηση αυθεντίας. Να τους πείσεις ότι αυτό που λες, το λες έχοντας την απαραίτητη "αυθεντία". Οι "επιστήμονες", ας πούμε, ή οι "ερευνητές", ακούνε τι λες και συμφωνούν μαζί σου. Η ο τάδε φιλόσοφος. Η το τάδε ιερό βιβλίο.

Δεν έχει σημασία το τι πραγματικά λένε όλοι αυτοί. Το βασικό είναι να πείσεις τον κόσμο ότι εσύ ξέρεις τι λένε και, ναι, συμφωνούν με σένα.

Όμως δεν αρκεί αυτό. Χρειάζεται κάτι ακόμα.

Χρειάζεται κάτι οικείο και καθησυχαστικό, αλλά με ένα μίνιμουμ έκπληξης. Μίνιμουμ όμως, όχι παραπάνω. Πρέπει να τους εκπλήξεις τόσο, όσο πατάει η γάτα. Πρέπει ωστόσο να στηρίζεσαι σε κάτι που έστω και εμμέσως είναι γνωστό και αποδεκτό.

Για παράδειγμα: οι Ελληνες είναι φιλόξενος και φιλικός λαός, άνθρωποι εργατικοί, τίμιοι, με φιλότιμο.

Αν θέλεις να σε πιστέψουν, απ' αυτό πρέπει να ξεκινήσεις. Όχι από τους τυφλούς της Ζακύνθου. Ασ' τους τους τυφλούς. Μην τους αναφέρεις καθόλου.

Μετά χρειάζεσαι την αυθεντία. Βάλε όποιον θέλεις. Τον Βαρουφάκη. Τον Κρούγκμαν. Τον Ζίζεκ. Τον Γκάντι. Τον Αριστοτέλη. Όποιον και να βάλεις δεν έχει σημασία. Σημασία έχει να πείσεις ότι ξέρεις τι κάνεις.

Και τέλος, χρειάζεσαι την έκπληξη.

Κι εδώ έρχεται ο Alexis.

Το παιδί των καταλήψεων. Ο φτωχός πλην τίμιος οραματιστής. Ο μορφωμένος, του πολυτεχνείου. Ο άθεος πλην ευσεβής. Ο επαναστάτης: αντισυμβατικός, έως και μπρουτάλ, αλλά με καθαρό χαμόγελο. Ο με χωρίς γραβάτα αλλά αδιάβλητος. Ο διαπραγματευτής. Το παιδί της διπλανής πόρτας που όλοι τον σέβονται. Κάθε μάνα τον θέλει για την κόρη της. Κάθε μπαμπάς τον θέλει για γιο.

Κι οι ξένοι; Ο Σόιμπλε, η Μέρκελ, ο Ολάντ. Σούζα όλοι μπροστά του. Ακόμα κι ο Ομπάμα.

Να τη η έκπληξη.

Και μετά απορούνε κάποιοι για το ΟΧΙ που έγινε ΝΑΙ.

Όχι παιδιά. Όσο υπάρχει ο Alexis δεν μας λείπει τίποτα.

Άλλωστε Alexis δεν υπάρχει. Μια λέξη είναι κι αυτός, ένα κενό σημαίνον. Ενας καθρέφτης. Μια λευκή οθόνη.

Μια οθόνη που σκορπίζει ηρωισμό και χαμόγελα επί όσο διαρκεί η προβολή.

Για μετά βλέπουμε.

Σας ασπάζομαι,
Θείος Ακάκιος.

σουξέ μέν



Αυτό το Mac mini είναι σχεδιασμένο ώστε να κάνει ό,τι θέλετε – και πολλά περισσότερα.

Αυτό το Mac mini δεν είναι το βασικό μοντέλο, και **κοστίζει 1249 ευρώ** (http://store.istorm.gr/mac/mac-mini/2-8ghz-mac-mini.html).

Αυτό το Mac mini το **κληρώνουμε μεταξύ 25 αναγνωστών** με Advanced Access.

Αν η συνδρομή σας πλησιάζει στη λήξη της, αυτό το Mac mini **είναι εξαιρετικά πιθανό** να γίνει δικό σας.

Για να πάρετε μέρος στην κλήρωση, **αρκεί να αποκτήσετε Advanced Access** συμπληρώνοντας τη φόρμα παραγγελίας στο http://deltahacker.gr/pdf12order/#s2.

Από την ίδια τη φόρμα παραγγελίας (http://deltahacker.gr/pdf12order/#s2) βλέπετε **πόσους αναγνώστες με Advanced Access χρειαζόμαστε ακόμα**, ώστε να προχωρήσουμε στην κλήρωση.

Καλή σας επιτυχία!

Skill: Beginner
Tags: VPN, OpenVPN, OpenVPN Access Server, OpenVPN Connect, no-brainer

OpenVPN Access Server, χωρίς βάσανα και πόνο!

Η παρουσία ενός OpenVPN server στο cloud αποτελεί άριστη λύση για την προστασία της ιδιωτικότητάς μας στο ίντερνετ. Όμως η σωστή ρύθμιση ενός τέτοιου server χρειάζεται μεγάλη προετοιμασία, τα επιμέρους βήματα σίγουρα δεν μπορούν να χαρακτηριστούν ως προφανή, ενώ γενικά το όλο πρότζεκτ δεν είναι ό,τι πιο απλό για τους νεοφερμένους στη γραμμή εντολών του Unix, στην κρυπτογραφία δημόσιου κλειδιού και στη λογική των Αρχών Πιστοποίησης. Ευτυχώς, τα πράγματα δεν χρειάζεται να είναι *τόσο* δύσκολα.

του Χρήστου Βαρελά

Σε αναλυτικό άρθρο που δημοσιεύεται στο site του περιοδικού δείχνουμε πώς στήνουμε έναν VPN server με τη βοήθεια του OpenVPN, σε ένα εικονικό ή φυσικό σύστημα με Ubuntu Server (http://deltahacker.gr/openvpn-revisited). Όπως ίσως θα γνωρίζετε, το λεγόμενο Community Edition του OpenVPN, αυτό δηλαδή με το οποίο καταπιανόμαστε στο προαναφερθέν άρθρο, παρέχεται χωρίς κανέναν περιορισμό. Η χρήση του όμως προϋποθέτει ένα σχετικά υψηλό επίπεδο τεχνογνωσίας, ενώ και η προετοιμασία καθώς και η διαχείριση του OpenVPN server δεν είναι εργασίες που στο Λευκαντί Ευβοίας θα χαρακτηρίζαμε ως no-brainers.

Για όλους όσοι επιθυμούν να ξεκινήσουν γρήγορα με τον δικό τους VPN server και ίσως ν' αφήσουν την κατανόηση της παρασκηνιακής λογικής γι' αργότερα, υπάρχει κι ο ωραιότατος OpenVPN Access Server. Πρόκειται για μια εκδοχή του OpenVPN η οποία εκτός από τον βασικό server παρέχει εύχρηστο web panel διαχείρισης, καθώς και μια σειρά από clients για Windows, OS X, Android και iOS (υπάρχει βεβαίως και δυνατότητα σύνδεσης από Linux, BSD και άλλα συστήματα Unix). Η OpenVPN Technologies Inc. (https://openvpn.net), η εταιρεία πίσω από τα Community Edition και Access Server του OpenVPN, τις δύο αυτές εκδοχές τις παρέχει δωρεάν αλλά στην Access Server επιτρέπει την ταυτόχρονη σύνδεση δύο μόνο *διαφορετικών* clients. Για περισσότερους ζητά την αγορά κατάλληλου license. Αν πάντως μιλάμε για μη-επαγγελματικό σενάριο χρήσης κι ενδιαφερόμαστε για περισσότερες από

δύο συσκευές (clients), υπάρχει τρόπος να μην πληρώσουμε: Αρκεί να έχουμε ένα λογαριασμό χρήστη για τον Access Server και γι' αυτόν να επιτρέψουμε τις ταυτόχρονες συνδέσεις, από διαφορετικές συσκευές. Η λύση σίγουρα δεν είναι ιδανική για ένα εταιρικό περιβάλλον, για μεμονωμένους χρήστες όμως είναι μια χαρά.

Στο υπόλοιπο του άρθρου δείχνουμε, αναλυτικά και βήμα προς βήμα, πώς εγκαθιστούμε και ρυθμίζουμε τον OpenVPN Access Server σε ένα VPS στο cloud, το οποίο τρέχει Ubuntu Server.

Προαπαιτούμενα

Χρειαζόμαστε ένα VPS στο cloud με Ubuntu Server, στο οποίο θα έχουμε πλήρη πρόσβαση μέσω SSH. Στο πλαίσιο του παρόντος άρθρου επιστρατεύσαμε ένα μικρό VPS από τη Digital Ocean με 512MB RAM, 20GB SSD drive και 1TB δωρεάν bandwidth ανά μήνα. Ως λειτουργικό για το droplet –έτσι ονομάζει η Digital Ocean τα VPS της– επιλέξαμε το Ubuntu Server 14.04 LTS 64bit. Η εταιρεία έχει data centers σε διάφορες γεωγραφικές περιοχές στον πλανήτη και το συγκεκριμένο droplet φιλοξενείται σε data center της Νέας Υόρκης. Το δε κόστος φιλοξενίας είναι $5 ανά μήνα κι αν αποφασίσετε να φτιάξετε και το δικό σας VPS στη Digital Ocean, ξεκινήστε από το http://bit.ly/digocean10off (referal URL) ώστε να κερδίσετε $10 έκπτωση σε credit. Πρακτικά, με το μικρό droplet θα έχετε δύο μήνες δωρεάν.

Φροντίσαμε εξάλλου να έχουμε πρόσβαση στο λογαριασμό ενός απλού χρήστη του VPS μας, μέσω SSH και χωρίς να πληκτρολογούμε password (http://deltahacker.gr/actsubs-openssh-secure-passwordless-logins). Ο εν λόγω χρήστης (sub0) έχει τη δυνατότητα να αποκτά δικαιώματα root, με το sudo. Το iptables το ρυθμίσαμε ώστε να επιτρέπονται, βεβαίως, οι συνδέσεις στο TCP port του OpenSSH server, ενώ είχαμε και το fail2ban ενεργό προκειμένου να αντιμετωπίζουμε τα brute force και DoS attacks προς την υπηρεσία του SSH. Περισσότερα για το πώς θα ρυθμίσετε το δικό σας VPS μπορείτε να διαβάσετε στο http://deltahacker.gr/actsubs-1srv2clouds-p23.

Σημειώστε ότι, ανεξαρτήτως του πώς θα ρυθμίσετε το iptables, μετά την εγκατάσταση του OpenVPN Access Server θα προστεθούν αυτόματα οι κατάλληλοι κανόνες ώστε να είναι εφικτή η πρόσβαση στο web panel, βεβαίως και στον ίδιο τον server. Επίσης, θα είναι δυνατή η αναδρομολόγηση της δικτυακής κίνησης των πελατών μέσω του απομακρυσμένου μηχανήματος.

Λήψη και εγκατάσταση του OpenVPN Access Server

Από τον αγαπημένο μας web browser μεταβαίνουμε στο δικτυακό τόπο του OpenVPN και συγκεκριμένα στη σελίδα

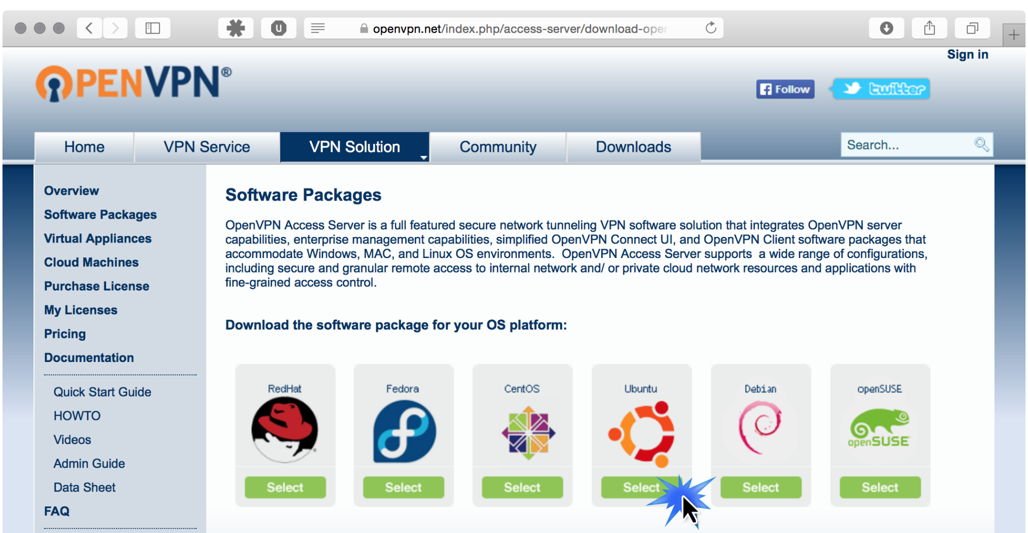
https://openvpn.net/index.php/access-server/overview.html

Αυτό που μας ενδιαφέρει είναι το κουμπί Access Server Software Packages, στ' αριστερά.



Για τον OpenVPN Access Server παρέχονται αυτόνομα πακέτα, κατάλληλα για εγκατάσταση σε μια πληθώρα διανομών Linux. Επιπρόσθετα, διατίθενται VMs και cloud servers με τον Access Server προεγκατεστημένο. Δεν σκοπεύουμε να εγκαταστήσουμε σε VM, ενώ έχουμε ήδη το VPS μας στο cloud. Κάνουμε, λοιπόν, ένα κλικ στο κουμπί Access Server Software Packages.

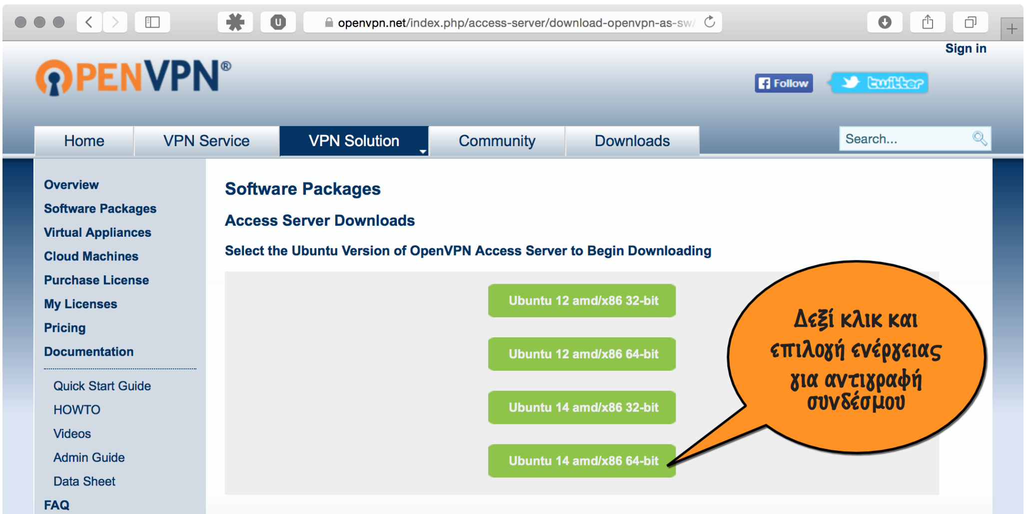
Πατάμε λοιπόν στο κουμπί Access Server Software Packages κι αμέσως φορτώνει η σελίδα για την επιλογή εκδοχής του Access Server, ανάλογα με τη διανομή Linux στην οποία σκοπεύουμε να τον εγκαταστήσουμε. Στο πλαίσιο της παρουσίασής μας ενδιαφερόμαστε για το Ubuntu.



Πακέτα εγκατάστασης του Access Server διατίθενται για όλες τις δημοφιλείς διανομές του Linux. Στο VPS μας τρέχουμε Ubuntu Server, οπότε πατάμε στο κουμπί Ubuntu.

Μετά από κλικ στο κουμπάκι του Ubuntu βλέπουμε ότι έχουμε τέσσερις επιλογές: 32 ή 64bit, για Ubuntu 12 ή 14. Στο VPS έχουμε το Ubuntu Server 14.04 LTS 64bit, οπότε η επιλογή που τώρα μας ενδιαφέρει είναι η τέταρτη: "Ubuntu 14 amd/x86 64-bit". Δεν θέλουμε βέβαια να κατεβάσουμε το αντίστοιχο πακέτο στον υπολογιστή από τον οποίο εργαζόμαστε, θέλουμε όμως το αντίστοιχο URL. Έτσι, κάνουμε δεξί κλικ πάνω στο "Ubuntu 14 amd/x86 64-bit" κι από το παράθυρο που εμφανίζεται επιλέγουμε την ενέργεια Copy Link ή Copy Link Address ή όπως αλλιώς λέγεται η ενέργεια αντιγραφής URL στον browser μας, τέλος πάντων. Στο πλαίσιο του παραδείγματος, το URL που μόλις αντιγράψαμε είναι το

http://swupdate.openvpn.org/as/openvpn-as-2.0.21-Ubuntu14.amd_64.deb



Στο VPS μας έχουμε Ubuntu Server 14.04 LTS 64bit, συνεπώς κάνουμε δεξί κλικ πάνω στο κουμπί "Ubuntu 14 amd/x86 64-bit" και επιλέγουμε τη λειτουργία για αντιγραφή URL. Το ίδιο το downloading θα γίνει σε λίγο, αφού πρώτα συνδεθούμε στο VPS.

Για τη συνέχεια συνδεόμαστε, μέσω SSH, στο VPS όπου πρόκειται να εγκατασταθεί ο Access Server. Κατεβάζουμε το σχετικό πακέτο πληκτρολογώντας

```
sub0@defiant:~$ wget http://swupdate.openvpn.org/as/openvpn-as-2.0.21-Ubuntu14.
amd_64.deb
```

Η εγκατάσταση του Access Server επιτυγχάνεται με τη βοήθεια του εργαλείου dpkg και με δικαιώματα root:

```
sub0@defiant:~$ sudo dpkg -i openvpn-as-2.0.21-Ubuntu14.amd_64.deb
```

```
cvar — sub0@defiant: ~ — ssh — 115×20 — ⌘1

sub0@defiant:~$
sub0@defiant:~$ wget http://swupdate.openvpn.org/as/openvpn-as-2.0.21-Ubuntu14.amd_64.deb
--2015-11-06 03:34:50--  http://swupdate.openvpn.org/as/openvpn-as-2.0.21-Ubuntu14.amd_64.deb
Resolving swupdate.openvpn.org (swupdate.openvpn.org)... 104.28.1.12, 104.28.0.12
Connecting to swupdate.openvpn.org (swupdate.openvpn.org)|104.28.1.12|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27002502 (26M) [application/octet-stream]
Saving to: 'openvpn-as-2.0.21-Ubuntu14.amd_64.deb'

100%[=========================================================================>] 27,002,502  85.0MB/s   in 0.3s

2015-11-06 03:34:51 (85.0 MB/s) - 'openvpn-as-2.0.21-Ubuntu14.amd_64.deb' saved [27002502/27002502]

sub0@defiant:~$
```

Δεν είναι κι άσχημη η ταχύτητα downstream του VPS μας, ε;

```
cvar — sub0@defiant: ~ — ssh — 115×20 — ⌘1
sub0@defiant:~$
sub0@defiant:~$ sudo dpkg -i openvpn-as-2.0.21-Ubuntu14.amd_64.deb
[sudo] password for sub0:
Selecting previously unselected package openvpn-as.
(Reading database ... 96719 files and directories currently installed.)
Preparing to unpack openvpn-as-2.0.21-Ubuntu14.amd_64.deb ...
Unpacking openvpn-as (2.0.21-Ubuntu14) ...
Setting up openvpn-as (2.0.21-Ubuntu14) ...
The Access Server has been successfully installed in /usr/local/openvpn_as
Configuration log file has been written to /usr/local/openvpn_as/init.log
Please enter "passwd openvpn" to set the initial
administrative password, then login as "openvpn" to continue
configuration here: https://104.131.181.237:943/admin
To reconfigure manually, use the /usr/local/openvpn_as/bin/ovpn-init tool.

Access Server web UIs are available here:
Admin  UI: https://104.131.181.237:943/admin
Client UI: https://104.131.181.237:943/
sub0@defiant:~$
```

Παρατηρώντας τα μηνύματα που εμφανίζονται κατά την εγκατάσταση του OpenVPS Access Server, παρατηρούμε ότι δημιουργείται ένας νέος λογαριασμός χρήστη με username το openvpn.

Η διαδικασία ολοκληρώνεται ταχύτατα και, μεταξύ άλλων, στο σύστημα δημιουργείται ένας νέος λογαριασμός χρήστη με username το openvpn. Πριν συνεχίσουμε οφείλουμε να ορίσουμε ένα ισχυρό password για τον εν λόγω λογαριασμό:

```
sub0@defiant:~$ sudo passwd openvpn
```

Καθορίστε ένα πανίσχυρο password. Το δικό μας, π.χ., είναι ένα εντελώς τυχαίο string μήκους 30 χαρακτήρων που συνδυάζει πεζοκεφαλαία, αριθμούς και σύμβολα. Είναι βεβαίως αδύνατο να το θυμόμαστε, οπότε το έχουμε αποθηκευμένο σε ένα ασφαλές μέρος (ως σημείωση στο LastPass).

cvar — sub0@defiant: ~ — ssh — 115×10 — ⌘1

```
sub0@defiant:~$
sub0@defiant:~$ sudo passwd openvpn
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
sub0@defiant:~$
```

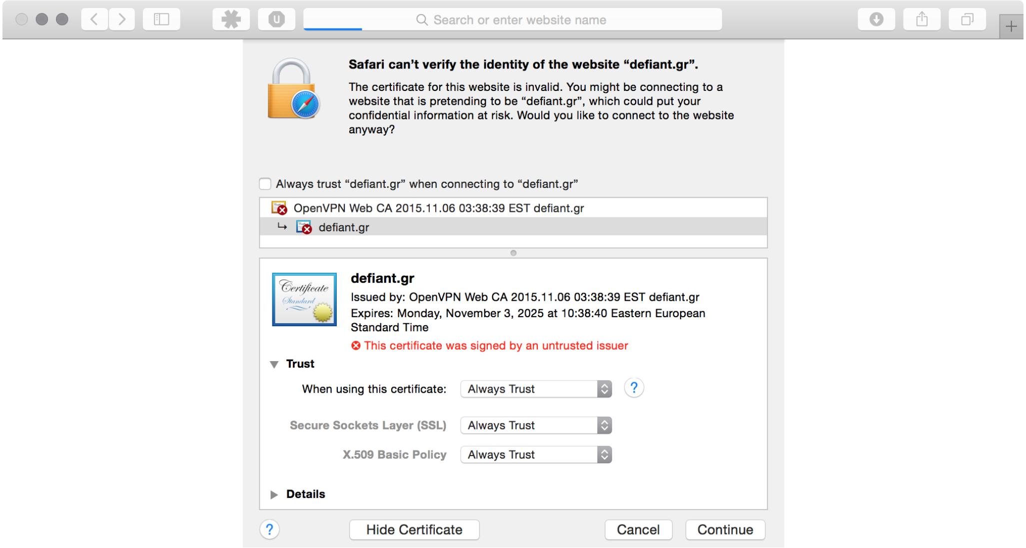
Για το λογαριασμό με όνομα openvpn οφείλουμε να ορίσουμε ένα *πανίσχυρο* password.

Πρώτη σύνδεση

Στη μπάρα διευθύνσεων του browser της προτίμησής μας πληκτρολογούμε

https://διεύθυνση_IP_του_VPS:943

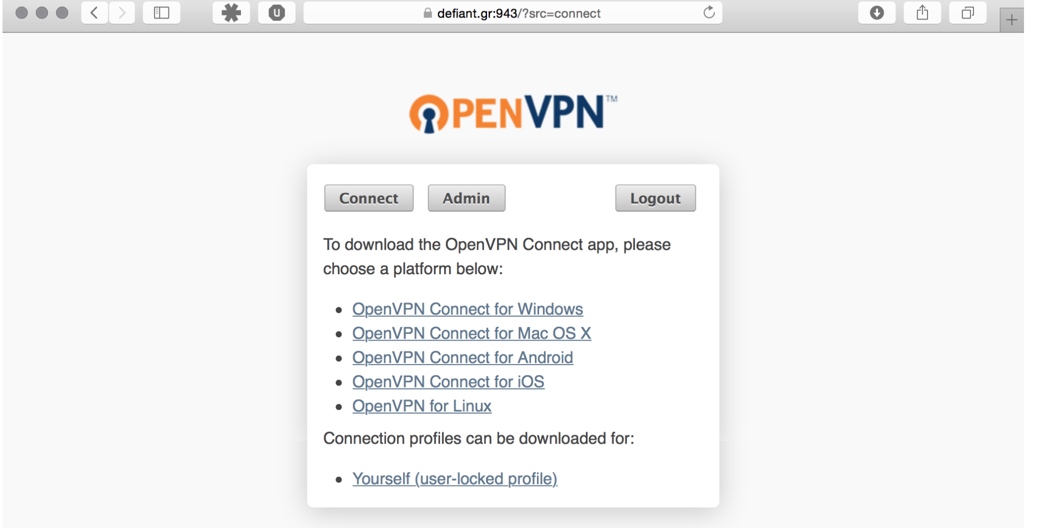
Προσοχή στο πρωτόκολλο, το οποίο πρέπει να είναι "https" κι όχι "http". Χρειάζεται επίσης να συμπεριλάβουμε το port, μετά την άνω-κάτω τελεία (943). Την πρώτη φορά που θα επισκεφτούμε το web panel του OpenVPN Access Server θα πάρουμε προειδοποίηση για self-signed πιστοποιητικό. Θα το δεχτούμε ως έγκυρο και θα προχωρήσουμε. Δείτε τα ακόλουθα screen shots και διαβάστε τις αντίστοιχες περιγραφές.



Το defiant.gr είναι το domain name του VPS στο οποίο έχουμε εγκατεστημένο τον OpenVPN Access Server μας. Αν δεν έχετε αντιστοιχισμένο domain name στο δημόσιο IP του VPS σας, θα δώσετε απευθείας το IP. Την πρώτη φορά που επιχειρούμε να επισκεφτούμε το web panel του Access Server παίρνουμε προειδοποίηση για self-signed πιστοποιητικό. Το δεχόμαστε ως έγκυρο και μάλιστα φροντίζουμε ώστε ο web browser να το εμπιστεύεται και στο μέλλον.



Μετά την αποδοχή του πιστοποιητικού πληκτρολογούμε το όνομα του χρήστη (openvpn) που δημιουργήθηκε κατά την εγκατάσταση του OpenVPN Access Server, καθώς και το password που καθορίσαμε πριν λίγο. Στο drop-down menu δίπλα από το "Go" επιλέγουμε το "Login" κι αμέσως κάνουμε κλικ στο κουμπί δεξιά.



Η απλότητα του OpenVPN Access Server σε όλο της το μεγαλείο! Προκειμένου να συνδεθούμε στον server αρκεί να κατεβάσουμε και να εγκαταστήσουμε τον κατάλληλο client για το λειτουργικό σύστημα του υπολογιστή ή της συσκευής που μας ενδιαφέρει. Δεν χρειαζόμαστε ούτε πιστοποιητικά, ούτε ιδιωτικά κλειδιά, ούτε παράξενες κι ακατανόητες ρυθμίσεις: Το μόνο που πρέπει να διαθέτουμε είναι το password του χρήστη openvpn, ώστε να το δώσουμε στον client όταν μας το ζητήσει.

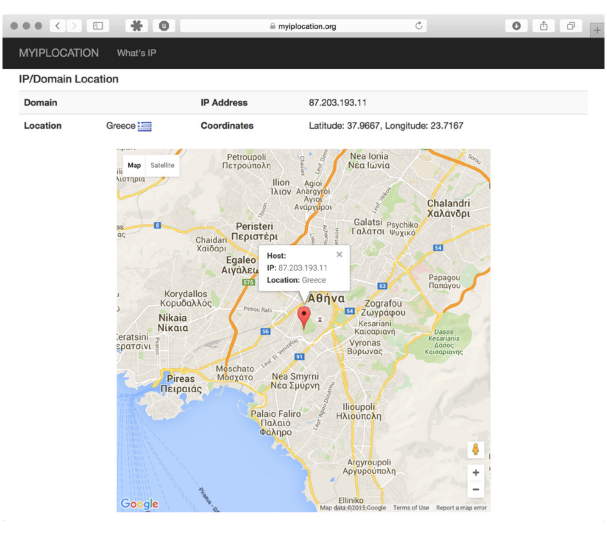
Η εγκατάσταση του λεγόμενου OpenVPN Connect client για υπολογιστές Windows ή OS X είναι απλούστατη και πραγματοποιείται με διαδοχικά κλικ στον αντίστοιχο installation wizard. Αναλόγως των ρυθμίσεων ασφαλείας στο OS X, σημειώστε ότι προκειμένου να εκτελεστεί ο installer ίσως χρειαστεί η παρέμβασή σας από το System Preferences → Security & Privacy → General → Allow apps downloaded from. Μετά την εγκατάσταση το πρόγραμμα θα είναι έτοιμο για να συνδεθεί στον απομακρυσμένο OpenVPN Access Server και το μόνο που θα πρέπει να γνωρίζουμε είναι το password του χρήστη με username το openvpn. Έχετε κατά νου ότι στα Windows καλό είναι να τρέχετε τον client με δικαιώματα διαχειριστή (δεξί κλικ πάνω στο σχετικό εικονίδιο κι επιλογή του Run as Administrator). Για τη σύνδεση εξάλλου από συσκευή με Android ή με iOS, κατεβάζουμε τη δωρεάν εφαρμογή OpenVPN Connect από το Play Store ή από το Apple Store αντίστοιχα. Εκτός από το password του χρήστη openvpn θα χρειαστούμε και το σωστό connection profile, το οποίο περιγράφεται σε ένα αρχείο ονόματι client.ovpn που λαμβάνουμε από τη σελίδα

https://διεύθυνση_IP_του_VPS:943/?src=connect

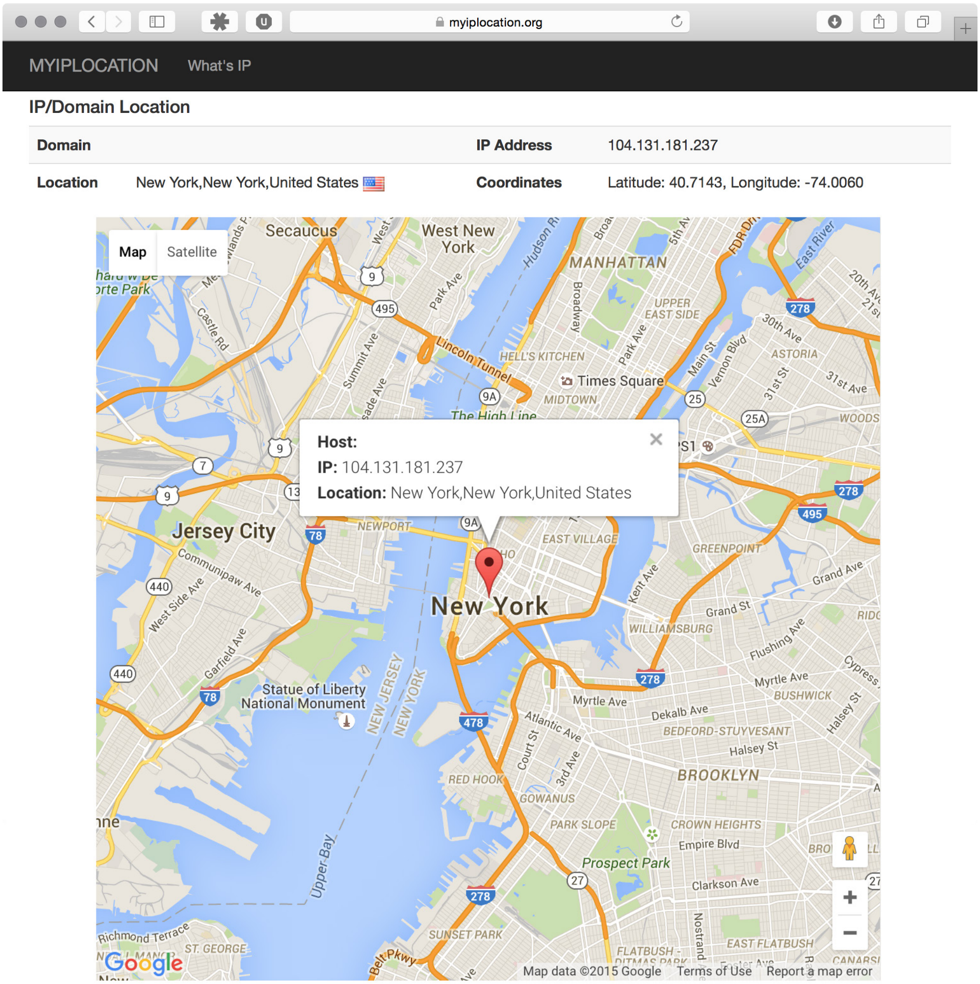
με ένα κλικ στο "Yourself (user-locked profile)". Το συγκεκριμένο αρχείο το κατεβάζουμε κι αν επιθυμούμε να συνδεθούμε στον Access Server από μηχάνημα με Linux ή όταν δεν θέλουμε να χρησιμοποιήσουμε το OpenVPN Connect αλλά ένα διαφορετικό πρόγραμμα σύνδεσης, όπως, π.χ., είναι το Viscosity (βλ. τελευταία ενότητα).

Δοκιμές πριν/μετά

Ένας γρήγορος τρόπος προκειμένου να βεβαιωθούμε ότι βγαίνουμε στο Internet μέσω του απομακρυσμένου OpenVPN Access Server κι όχι μέσω του ISP μας, είναι με δύο διαδοχικές επισκέψεις σε κάποιο από τα sites που αναφέρουν το δημόσιο IP καθώς και τη γεωγραφική μας θέση. Ένα εξ αυτών είναι το https://myiplocation.org.

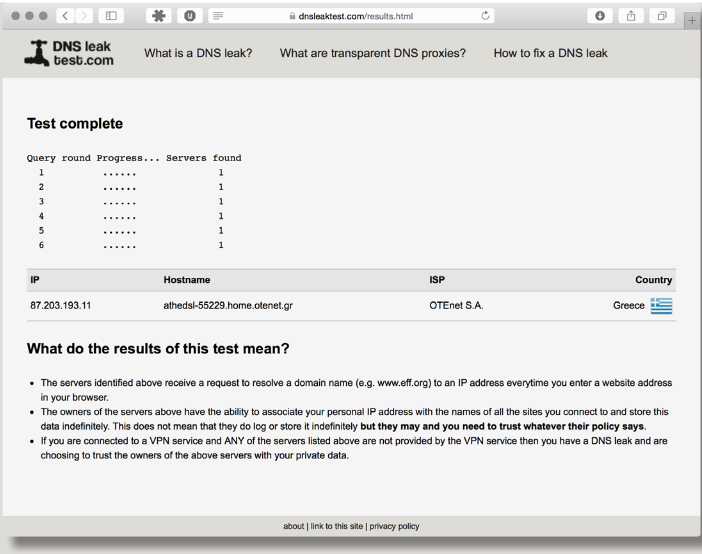


Χωρίς να συνδεθούμε στον Access Server, από έναν web browser επισκεφτόμαστε τη διεύθυνση https://myiplocation.org. Παρατηρήστε τη διεύθυνση IP καθώς και τη γεωγραφική θέση. Η δεύτερη δεν ανταποκρίνεται στην πραγματικότητα, τουλάχιστον όμως είναι εντός της χώρας.

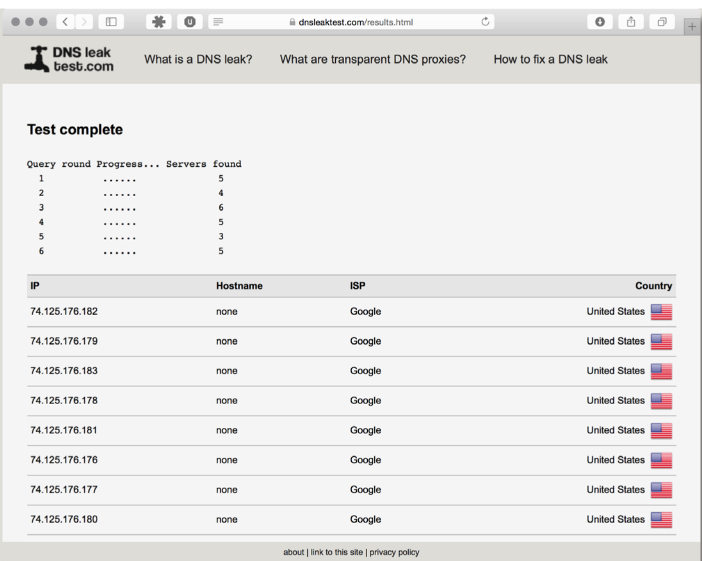


Αφού συνδεθούμε στον Access Server πηγαίνουμε ξανά στο https://myiplocation.org. Η διεύθυνση IP έχει αλλάξει, το ίδιο και η γεωγραφική μας θέση: για το Internet, φαίνεται ότι έχουμε αμερικάνικη διεύθυνση IP και βρισκόμαστε στο Μανχάταν.

Εξ ορισμού, για όσο είμαστε συνδεδεμένοι στον OpenVPN Access Server όλα τα DNS queries περνάνε μέσα από το κρυπτογραφημένο κανάλι που ξεκινά από τον υπολογιστή μας και φτάνει έως το VPS. Έτσι, ο ISP μας αφενός δεν βλέπει το περιεχόμενο των δικτυακών πακέτων που στέλνουμε και λαμβάνουμε (είναι κρυπτογραφημένα), αφετέρου δεν έχει ιδέα για τα sites και τις υπηρεσίες που επισκεπτόμαστε ή προσπελάζουμε. Για να δούμε πώς αλλάζουν οι χρησιμοποιούμενοι name servers *μετά* τη σύνδεσή μας στον Access Server, μπορούμε να κάνουμε δύο διαδοχικές επισκέψεις στο δικτυακό τόπο http://dnsleaktest.com και να τρέξουμε το Standard ή το Extended test.



Μπροστά από το laptop των δοκιμών μας υπάρχει ένας pfSense router που παρέχει υπηρεσίες recursive DNS στα μηχανήματα του τοπικού δικτύου (http://deltahacker.gr/actsubs-how-dns-works-p4). Πριν συνδεθούμε στον OpenVPN Access Server μεταβαίνουμε στο http://dnsleaktest.com και τρέχουμε το Extended test. Μετά από λίγο φαίνεται ότι ο name server είναι το μηχάνημα με δημόσιο IP αυτό που έχουμε πάρει από τον ISP μας. Το αποτέλεσμα είναι αναμενόμενο, αφού ο name server του pfSense είναι recursive.



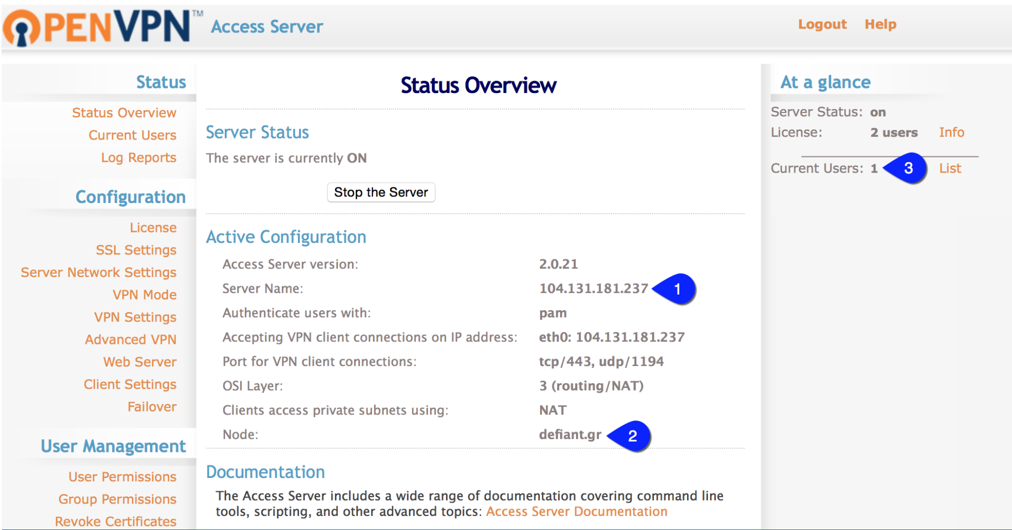
Φροντίζουμε ώστε να συνδεθούμε στον Access Server, μεταβαίνουμε ξανά στο http://dnsleaktest.com και για άλλη μια φορά τρέχουμε το Extended test. Μερικά δευτερόλεπτα αργότερα διαπιστώνουμε ότι, πλέον, χρησιμοποιούμε μια σειρά από name servers της Google στην Αμερική. Πρόκειται για τους name servers που γνωρίζει το VPS μας στη Digital Ocean και τους οποίους ο OpenVPN Access Server φρόντισε να "περάσει" στον client – εν προκειμένω στο laptop των δοκιμών.

Το web panel διαχείρισης

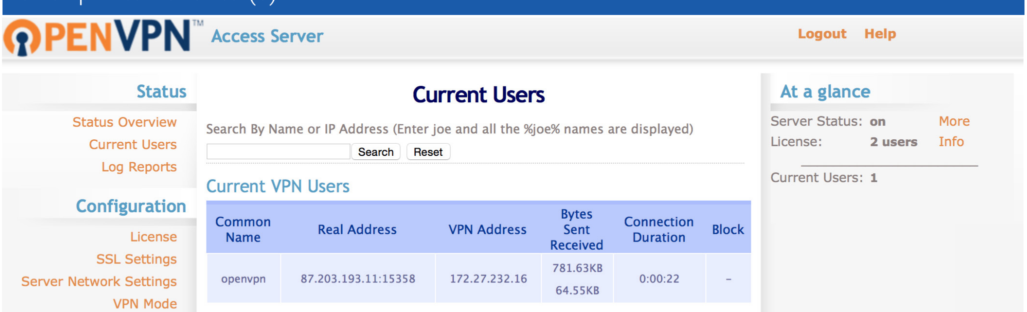
Η ρύθμιση του OpenVPN Access Server δεν χρειάζεται να γίνεται με την τροποποίηση configuration files, αλλά μέσω του παρεχόμενου web panel. Για απλή καθημερινή χρήση το πιθανότερο είναι ότι δεν θα χρειαστεί να επέμβετε στη λειτουργία του server. Καλό είναι πάντως να έχετε μια ιδέα για το περιβάλλον του web panel. Από έναν browser μεταβείτε στην ακόλουθη διεύθυνση

https://διεύθυνση_IP_του_VPS:943/admin

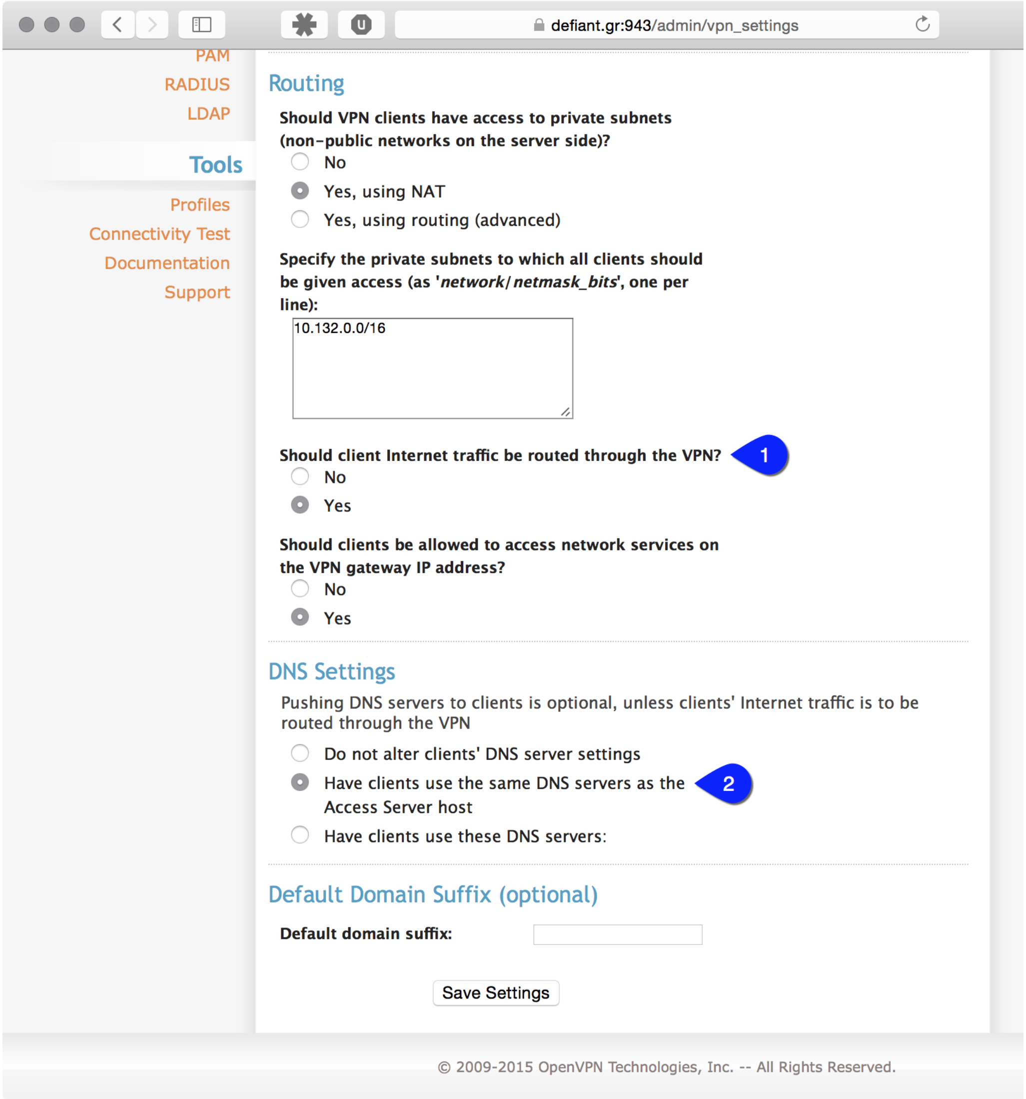
και συνδεθείτε στο λογαριασμό με username το openvpn. Ρίξτε μια ματιά στα ακόλουθα screenshots, διαβάστε και τις αντίστοιχες περιγραφές.



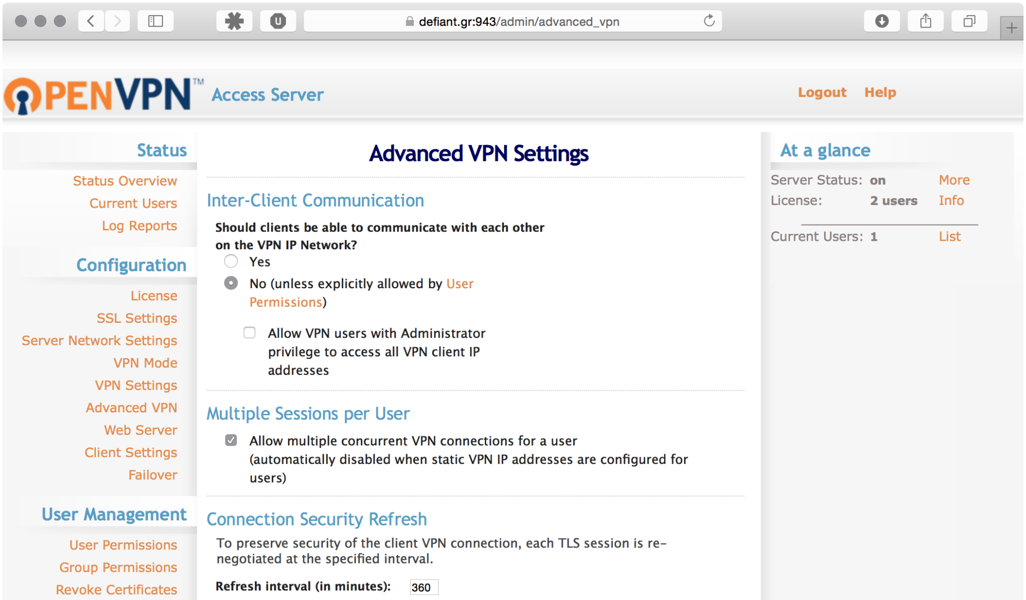
Στη σελίδα Status Overview βλέπουμε, μεταξύ άλλων, το δημόσιο IP του OpenVPN Access Server (1), το domain name (αν υπάρχει) (2), καθώς και το πλήθος των συνδεδεμένων clients (3).



Ιδού περισσότερες λεπτομέρειες για τον έναν και μοναδικό πελάτη που αυτή τη στιγμή είναι συνδεδεμένος στον OpenVPN Access Server. Παρατηρήστε ότι καταγράφεται και η δημόσια διεύθυνση IP που ο πελάτης έχει πάρει από τον ISP. Στο ίντερνετ, πάντως, βγαίνει με το δημόσιο IP του VPS στο οποίο κατοικεί ο Access Server.



Στις ρυθμίσεις περί Routing δείτε ότι, εξ ορισμού, η διαδικτυακή κίνηση των πελατών δρομολογείται μέσω του OpenVPN Access Server (1). Σε διαφορετική περίπτωση οι πελάτες ναι μεν θα αποκτούσαν ασφαλή πρόσβαση στο δίκτυο του απομακρυσμένου server, αλλά θα εξακολουθούσαν να βγαίνουν στο ίντερνετ μέσω του ISP. Προσέξτε εξάλλου κι αυτό το DNS Setting, λίγο πιο κάτω: Οι πελάτες είναι δυνατόν **α)** να συνεχίσουν να χρησιμοποιούν τους name servers που χρησιμοποιούσαν πριν τη σύνδεση, **β)** να αρχίσουν να χρησιμοποιούν τους name servers τους οποίους γνωρίζει ο Access Server (προεπιλογή) (2) ή **γ)** ν' αρχίσουν να χρησιμοποιούν τους name servers που εμείς καθορίζουμε.



Στη σελίδα Advanced VPN Settings υπάρχει μια εξαιρετικά χρήσιμη επιλογή, η οποία είναι εξ ορισμού ενεργοποιημένη. Πρόκειται για την Multiple Sessions per User: Χάρη σ' αυτή επιτρέπονται περισσότερες από μία συνδέσεις στον ίδιο λογαριασμό του Access Server, συνεπώς δεν χρειάζεται να αγοράσουμε license προκειμένου να συνδέσουμε περισσότερους από έναν υπολογιστές ή συσκευές. Η συγκεκριμένη προσέγγιση δεν είναι ιδανική για εταιρικά περιβάλλοντα, είναι όμως ό,τι πρέπει για όσους εκμεταλλεύονται τον Access Server για προσωπική χρήση.

Σύνδεση από Linux κι από δικό μας client

Για τη σύνδεση στον OpenVPN Access Server από Linux, αρκεί να υπάρχει στη διανομή εγκατεστημένο το πακέτο του OpenVPN Community Edition. Στο Ubuntu, π.χ., το εγκαθιστούμε δίνοντας

```
sudo apt-get install openvpn
```

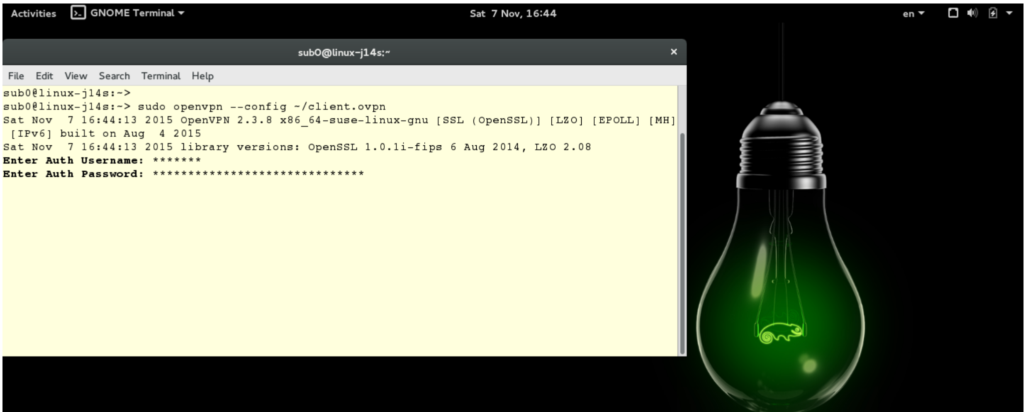
ενώ στο openSUSE είναι ήδη εγκατεστημένο. Το συγκεκριμένο πακέτο περιλαμβάνει το ομώνυμο εκτελέσιμο του OpenVPN, το οποίο μπορεί να λειτουργεί είτε ως server είτε ως client. Μας ενδιαφέρει η λειτουργία client, οπότε κατεβάζουμε το connection profile (αρχείο client.ovpn) από το

https://διεύθυνση_IP_του_VPS:943/?src=connect

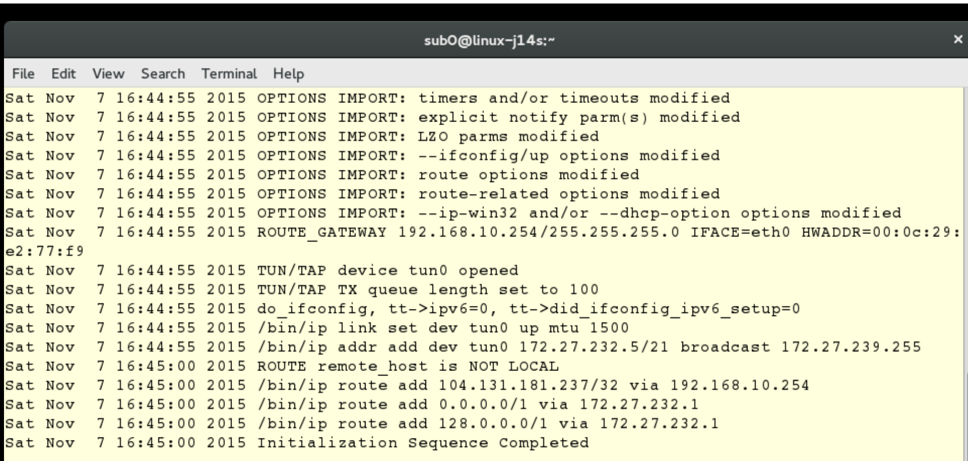
και σ' ένα τερματικό πληκτρολογούμε

```
sudo openvpn --config ~/client.ovpn
```

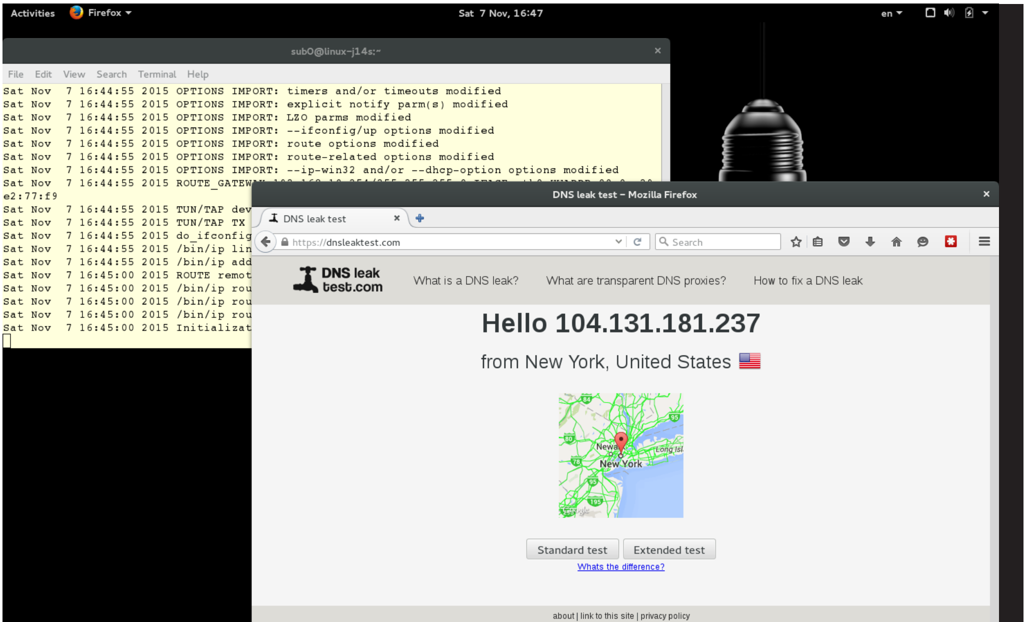
(υποθέσαμε ότι το αρχείο client.ovpn είναι στον προσωπικό κατάλογο του χρήστη). Δείτε λίγο τα ακόλουθα screen shots.



Βρισκόμαστε στο περιβάλλον GNOME του openSUSE Leap κι έχουμε ήδη κατεβάσει το αρχείο client.ovpn, δηλαδή το προκαθορισμένο connection profile προς το λογαριασμό του χρήστη openvpn, στον απομακρυσμένο OpenVPN Access Server. Η σύνδεση επιτυγχάνεται από ένα οποιοδήποτε τερματικό και με δικαιώματα root. Παρατηρήστε ότι χρειάζεται να πληκτρολογήσουμε τα username και password του απομακρυσμένου χρήστη.



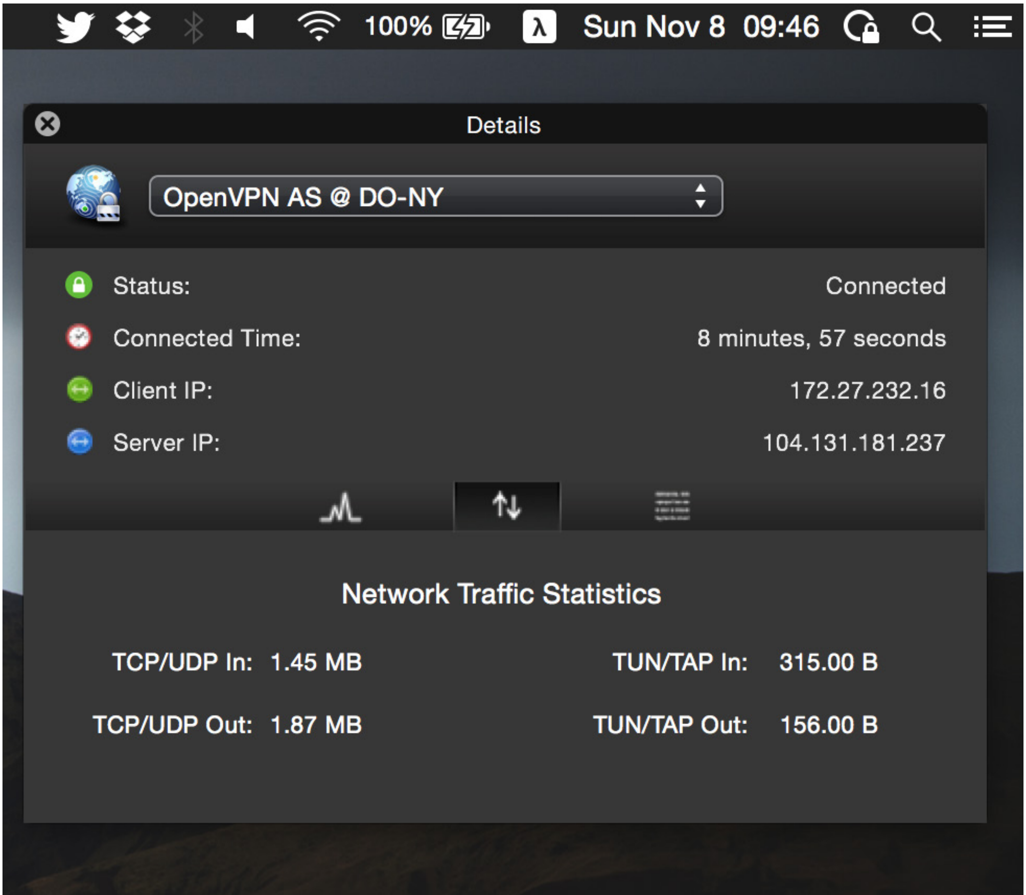
Τα μηνύματα στο τερματικό δείχνουν την πρόοδο της (σύντομης) διαδικασίας σύνδεσης και το τελευταίο φανερώνει ότι επετεύχθη επιτυχώς.



Με μια γρήγορη επίσκεψη στο https://dnsleaktest.com διαπιστώνουμε ότι για το ίντερνετ έχουμε αμερικάνικο IP – κι ας βρισκόμαστε κάπου στην Καλαμαριά.

Το αρχείο cilent.ovpn δεν πρέπει να διαρρεύσει. Αν κάποιος τρίτος το πάρει και με κάποιον τρόπο μάθει και το password του χρήστη openvpn στον απομακρυσμένο OpenVPN Access Server, τότε θα είναι κι εκείνος σε θέση να συνδέεται. Το εν λόγω αρχείο είναι χρήσιμο κι όταν αντί για τον OpenVPN Connect client θέλουμε να χρησιμοποιήσουμε κάποιον άλλον, όπως, π.χ., είναι το Viscosity (https://www.sparklabs.com/viscosity, για OS X και Windows). Το προγραμματάκι δεν διανέμεται δωρεάν, σε σύγκριση όμως με το OpenVPN Connect το βρίσκουμε πολύ πιο λειτουργικό κι ευέλικτο.

Χαρείτε τον ολοκαίνουργιο VPN server σας κι εκμεταλλευτείτε τον, ειδικά από το laptop σας κι άλλες φορητές συσκευές. Σε περίπτωση που ενδιαφέρεστε να ενισχύσετε την ιδιωτικότητα *ολόκληρου* του οικιακού σας τοπικού δικτύου κι όχι μεμονομένων υπολογιστών ή/και συσκευών σας, διαβάστε το άρθρο που ξεκινά από τη σελίδα 70 του παρόντος τεύχους.



Το Viscosity, για OS X και Windows, είναι ο αγαπημένος μας VPN client. Μεταξύ άλλων είναι σε θέση να καταλαβαίνει και τα connection profiles του OpenVPN Access Server.



Skill: Intermediate
Tags: frame, scanline, HSYNC, VSYNZ, pixel clock, video blanking

Κατασκευάστε έναν υπολογιστή, μέρος 3

Pixel προς pixel!

Έφτασε η ώρα να μελετήσουμε το firmware της κάρτας γραφικών του υπολογιστή μας. Σε αυτό το άρθρο θα γνωρίσουμε όλα τα χαρακτηριστικά και τις ιδιοτροπίες του σήματος VGA, ενώ θα εξετάσουμε και όλες τις προγραμματιστικές τεχνικές που επιστρατεύσαμε για τη σύνθεσή του.

του Παναγιώτη Βαρελά

Στο προηγούμενο μέρος της σειράς (http://deltahacker.gr/actsubs-byopc-p2) μελετήσαμε τα κυκλώματα που απαρτίζουν τον υπολογιστή μας. Αυτό δεν σημαίνει ότι ξεμπερδέψαμε με την κατασκευή. Ο δικός μας υπολογιστής δεν χρησιμοποιεί εξειδικευμένα τσιπ για την εικόνα και τον ήχο. Το ίδιο ισχύει και για τον κεντρικό επεξεργαστή, ο οποίος λειτουργεί ως interpreter της BASIC και ταυτόχρονα ως ένα υπεραπλουστευμένο "τσίπσετ". Σε κάθε περίπτωση έχουμε επιστρατεύσει έναν μικροελεγκτή της οικογένειας ATMEGA. Αυτό σημαίνει ότι ο υπολογιστής μας δεν μπορεί να ενεργοποιηθεί αμέσως μετά τη συναρμολόγηση των κυκλωμάτων. Για να λειτουργήσει κάθε υποσύστημα με τον προβλεπόμενο τρόπο, πρέπει πρώτα να προγραμματίσουμε κατάλληλα τους μικροελεγκτές.

Σε αυτό το άρθρο θα ξεκινήσουμε με το firmware της "κάρτας γραφικών". Εξετάζοντας το σχετικό κύκλωμα, αναφέραμε ότι τα δεδομένα που συνθέτουν την εικόνα αποθηκεύονται σ' ένα τσιπάκι μνήμης (frame buffer). Είδαμε επίσης ότι στο ρόλο της GPU βρίσκεται ένας ATMEGA644. Ε, λοιπόν, το πρόγραμμα που γράψαμε για αυτόν τον μικροελεγκτή αξιοποιεί τα δεδομένα του frame buffer και σχηματίζει το σήμα που οδηγεί την οθόνη. Αργότερα θα δούμε ότι το ίδιο πρόγραμμα εκτελεί πολλές ακόμα λειτουργίες, αλλά η παραγωγή του σήματος εικόνας είναι μακράν η κρισιμότερη. Το σήμα που παράγεται είναι τύπου VGA και το προτιμήσαμε για δύο λόγους. Αφενός, τα υπόλοιπα αναλογικά σήματα είναι αρκετά παλιά κι έχουν αρχίσει να εκλείπουν από τις σύγχρονες συσκευές. Αν επιλέγαμε κάτι σε CGA, EGA ή ακόμα και Composite, θα περιορίζαμε δραστικά το πλήθος των συμβατών οθονών. Αφετέρου, τα ψηφιακά σήματα απαιτούν αρκετά ισχυρότερο υλικό και η παραγωγή τους αποτελεί άπιαστο όνειρο για τους 8μπιτους AVR. Την ίδια στιγμή, όλες οι οθόνες και οι τηλεοράσεις των τελευταίων δεκαετιών διαθέτουν τουλάχιστον μια είσοδο VGA.

Εξελικτικό απομεινάρι

Σας λέει τίποτα η λέξη "πυροβόλο", εκτός από το γνωστό όπλο; Τι καταλαβαίνετε όταν ακούτε ότι "πρέπει να απομαγνητίσουμε τα πηνία"; Αν νομίζετε ότι ο γράφων άρχισε να παραμιλά, φταίει που δεν εξοικειωθήκατε ποτέ με τις οθόνες CRT. Το όνομά τους προκύπτει από το σπουδαιότερο εξάρτημα του μηχανισμού που σχημάτιζε την εικόνα. Αναφερόμαστε στην καθοδική λυχνία (Cathode Ray Tube) που οι παλιότεροι ονόμαζαν και "πυροβόλο". Το συγκεκριμένο εξάρτημα εξέπεμπε μια δέσμη ηλεκτρονίων, η οποία μπορούσε να εκτραπεί οριζοντίως και καθέτως με τη βοήθεια τεσσάρων πηνίων (δύο για την οριζόντια εκτροπή και δύο για την κάθετη). Η εν λόγω δέσμη λειτουργούσε σαν ένα λεπτεπίλεπτο πινέλο, που σάρωνε την επιφάνεια της οθόνης και σχημάτιζε την εικόνα. Η σάρωση ξεκινούσε από την πάνω αριστερή γωνία (όπως κοιτάμε εμείς την οθόνη), ακολουθούσε μια πορεία ζιγκ-ζαγκ κι έφτανε μέχρι την κάτω δεξιά γωνία. Με αυτόν τον τρόπο η εικόνα σχηματιζόταν σταδιακά, γραμμή προς γραμμή. Με την ευκαιρία, σημειώστε ότι κάθε τέτοια γραμμή ονομάζεται scanline, ενώ το σύνολο των γραμμών ονομάζεται frame ή field. Φυσικά, όλα τα παραπάνω εξελίσσονταν με μεγάλη ταχύτητα και δεν γίνονταν αντιληπτά από τις αισθήσεις μας. Αν εξαιρέσουμε ένα αμυδρό τρεμόπαιγμα, που κι αυτό δεν εντοπιζόταν πάντα, τα μάτια μας έβλεπαν μόνο την ολοκληρωμένη εικόνα. Μήπως αναρωτιέστε τι μας έπιασε τώρα και το ρίξαμε στην αρχαιολογία; Ο τρόπος λειτουργίας των παλιών τηλεοράσεων καθόρισε σε μεγάλο βαθμό την "εσωτερική δομή" των αναλογικών σημάτων εικόνας: Όλα τα σήματα του είδους περιλαμβάνουν έναν παλμό κατακόρυφου συγχρονισμού που σηματοδοτεί την έναρξη των frames, έναν παλμό οριζόντιου συγχρονισμού που σηματοδοτεί την εκκίνηση των scanlines και, στο ενδιάμεσο, τη χρωματική πληροφορία για τα διαδοχικά pixels που απαρτίζουν την εκάστοτε γραμμή.



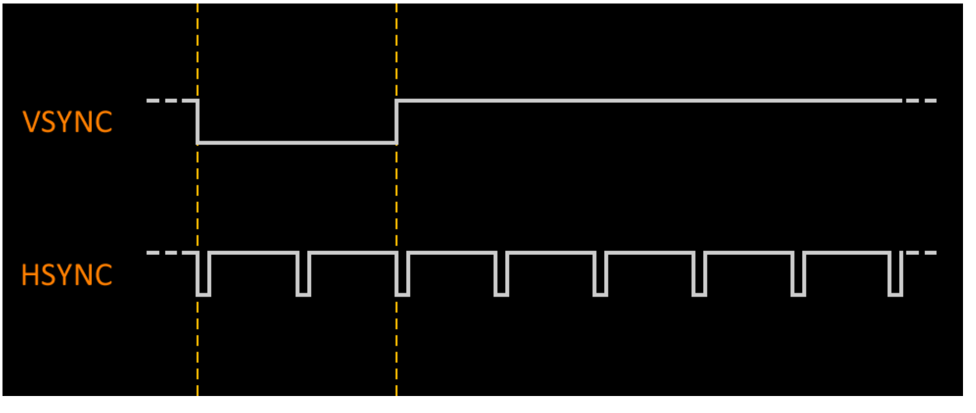
Στο πίσω άκρο της οθόνης φαίνεται η καθοδική λυχνία (Cathode Ray Tube), που παρήγαγε τη δέσμη ηλεκτρονίων. Λίγο πιο μπροστά διακρίνονται τα πηνία της κατακόρυφης και οριζόντιας εκτροπής της δέσμης. Αυτός ο πρωτόγονος μηχανισμός έχει αφήσει τη στάμπα του σε όλα τα αναλογικά σήματα εικόνας.

Λαβύρινθος προδιαγραφών

Αν και τα κυκλώματα VGA (Video Graphics Array) εμφανίστηκαν για πρώτη φορά το 1987, στους υπολογιστές IBM PS/2, το όνομα VGA χρησιμοποιείται ευρύτατα μέχρι σήμερα και κατέληξε να σημαίνει πολλά διαφορετικά πράγματα. Για παράδειγμα, VGA ονομάζεται το αναλογικό σήμα εικόνας που παράγουν οι κάρτες γραφικών, όπως επίσης και η υποδοχή D-sub 15 που έχει καθιερωθεί για τη μεταφορά του εν λόγω σήματος. Εξάλλου, με το ίδιο όνομα γίνεται λόγος και για τις ίδιες τις κάρ-

τες γραφικών. Το μεγάλο μπέρδεμα όμως σχετίζεται με τις αναλύσεις. Το όνομα VGA αναφέρεται και στην ανάλυση που προσέφεραν τα πρώτα κυκλώματα του είδους (640 x 480 pixels). Για τις υπόλοιπες αναλύσεις --ακόμα κι όταν μεταφέρονται με αναλογικό σήμα VGA--, χρησιμοποιούνται διαφορετικά ονόματα (SVGA, SXGA, WUXGA, κ.λπ.). Στη δική μας κατασκευή, πάντως, αποφασίσαμε να υιοθετήσουμε την παραδοσιακή ερμηνεία του όρου. Έτσι, το κύκλωμα γραφικών του υπολογιστή μας παράγει ένα σήμα, που έχει τα ίδια βασικά χαρακτηριστικά με το σήμα των πρώτων κυκλωμάτων VGA.

Όπως όλα τα αναλογικά σήματα, το σήμα VGA περιλαμβάνει έναν παλμό με το όνομα VSYNC (κατακόρυφος συγχρονισμός) και έναν παλμό ονόματι HSYNC (οριζόντιος συγχρονισμός). Επιπρόσθετα, καθώς το πρότυπο VGA προβλέπει τη χρήση χρωμάτων, υπάρχουν τρεις ακόμα "γραμμές", που μεταφέρουν την πληροφορία για κάθε χρωματική συνιστώσα (Red, Green και Blue). Τα σήματα VSYNC και HSYNC είναι ψηφιακά και "ανεστραμμένης" ή "αρνητικής" λογικής. Αυτό σημαίνει ότι βρίσκονται μονίμως στη λογική κατάσταση "HIGH" και μεταβαίνουν στην κατάσταση "LOW" προσωρινά, για όσο χρόνο είναι ενεργοί οι αντίστοιχοι παλμοί. Τα άλλα τρία σήματα είναι αναλογικά και η τάση του καθενός κυμαίνεται στο διάστημα από 0 έως 0,7Volt. Το άνω άκρο ερμηνεύεται ως πλήρη φωτεινότητα για την εκάστοτε συνιστώσα, ενώ το κάτω ερμηνεύεται ως μηδενική φωτεινότητα. Η παραπάνω περιγραφή είναι απόλυτα ακριβής και ισχύει για όλα τα σήματα VGA, αλλά δεν αρκεί για την υλοποίησή τους. Για τη σύνθεση ενός τέτοιου σήματος πρέπει να γνωρίζουμε τα λεγόμενα timings. Με απλά λόγια, πρέπει να γνωρίζουμε πόσο διαρκεί κάθε παλμός και με ποια συχνότητα επαναλαμβάνεται, όπως επίσης και το χρόνο που εκπέμπεται η χρωματική πληροφορία για κάθε pixel. Όλα αυτά τα μεγέθη καθορίζονται από το ρυθμό ανανέωσης της εικόνας και, προφανώς, από την ανάλυση.



Ο παλμός οριζόντιου συγχρονισμού επαναλαμβάνεται στην εκκίνηση κάθε scanline. Ο παλμός κατακόρυφου συγχρονισμού επαναλαμβάνεται ανά 525 scanlines (στην εκκίνηση κάθε frame) και διαρκεί όσο δύο ολόκληρα scanlines.

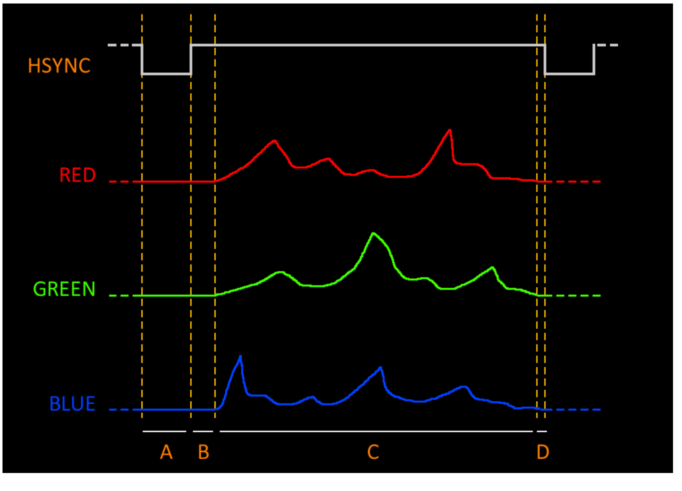
Η τυπική ανάλυση VGA ανέρχεται στα 640 x 480 pixels και ο αντίστοιχος ρυθμός ανανέωσής της (refresh rate) φτάνει στα 59,94Hz. Αυτό σημαίνει ότι τα δεδομένα που συνθέτουν την εικόνα αποστέλλονται στην οθόνη 59,94 φορές το δευτερόλεπτο και, συνεπώς, ο παλμός VSYNC επαναλαμβάνεται με συχνότητα 59,94Hz. Μη βιαστείτε να κάνετε διαιρέσεις, ώστε να υπολογίσετε τους χρόνους των υπολοίπων σημάτων. Υπάρχει μια λεπτομέρεια στη δομή του σήματος VGA που θα μπορούσε άνετα να θεωρηθεί ως παγίδα και πηγάζει (από πού αλλού;) από την τεχνολογία των οθονών CRT. Επειδή η συμπεριφορά των πηνίων παρουσιάζει αδράνεια, κάθε φορά που η δέσμη ηλεκτρονίων έφτανε στο τέλος μιας γραμμής δεν μπορούσε να φρενάρει και να επιστρέψει ακαριαία στη θέση εκκίνησης της επόμενης γραμμής. Το ίδιο ίσχυε και στον τερματισμό της σάρωσης του frame (κάτω δεξιά), οπότε η δέσμη έπρεπε να επιστρέψει στο σημείο έναρξης (πάνω αριστερά). Αυτή η αλλαγή στην κατεύθυνση κίνησης, όπως και η ίδια η μετατόπιση, απαιτούσε κάποιο χρόνο. Ακριβώς γι' αυτό, το σήμα VGA με ανάλυση 640 x 480 περιλαμβάνει 525 scanlines και όχι 480, όπως θα περίμενε κανείς. Οι πρόσθετες γραμμές ορίζουν το λεγόμενο vertical video blanking interval και δεν περιλαμβάνουν κάποια ορατή πληροφορία για την εικόνα. (Στην περίπτωση του αναλογικού τηλεοπτικού σήματος, οι συγκεκριμένες γραμμές χρησιμοποιούνταν για την εκπομπή κάποιων metadata.) Η ύπαρξη αυτών των γραμμών αποσκοπεί μόνο στην εισαγωγή μιας μικρής καθυστέρησης, ώστε να επιστρέψει η δέσμη ηλεκτρονίων στη θέση έναρξης του frame. Αντίστοιχα, κάθε scanline περιλαμβάνει 800 pixel και όχι 640. Τα πρόσθετα pixel δεν είναι ορατά και ορίζουν το λεγόμενο horizontal video blanking interval. Η παρουσία τους επιτρέπει στη δέσμη ηλεκτρονίων να μεταβεί στη θέση εκκίνησης της επόμενης γραμμής. Αυτά τα "νεκρά" χρονικά διαστήματα χωρίζονται σε δύο επιμέρους τμήματα και έχουν τα ονόματα back porch και front porch. Το back porch, σε κάθε scanline, βρίσκεται ακριβώς πριν από τα 640 ορατά pixels, ενώ το front porch ακολουθεί. Αντίστοιχα, το back porch σε κάθε frame βρίσκεται πριν από τις 480 ορατές γραμμές, ενώ το front porch ακολουθεί. Τελικά, το κλασικό σήμα VGA --αυτό που παράγει και το δικό μας κύκλωμα γραφικών-- έχει τα χαρακτηριστικά που φαίνονται στους ακόλουθους πίνακες.

Χρόνοι για κάθε scanline

	Pixels	Χρόνος [μsec]	Περιοχή
Παλμός HSYNC	96	3,8133	horizontal blanking
Back porch	40	1,5889	
Border left	8	0,3178	horizontal visible
Ορατό τμήμα γραμμής	640	25,4220	
Border right	8	0,3178	
Front porch	8	0,3178	horizontal blanking
πλήρες scanline	800	31,7776	

Χρόνοι για κάθε frame			
	Scanlines	Χρόνος [msec]	Περιοχή
Παλμός VSYNC	2	0,0636	vertical blanking
Back porch	25	0,7945	
Border top	8	0,2542	vertical visible
Ορατό τμήμα frame	480	15,2532	
Border bottom	8	0,2542	
Front porch	2	0,0636	vertical blanking
Ολόκληρο frame	525	16,6833	

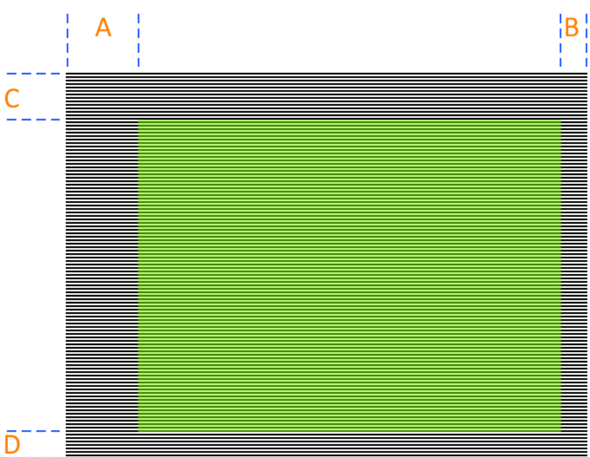
Αν κάνετε τις πράξεις θα καταλήξετε στα εξής: Ο χρόνος για κάθε pixel αγγίζει μόλις τα 39,72nsec, που σημαίνει ότι η συχνότητα μετάδοσης των δεδομένων των pixel (pixel clock) ανέρχεται στα 25,175MHz. Αυτά τα μεγέθη, όπως και όσα περιλαμβάνονται στους δύο πίνακες, δεν μας απασχόλησαν στο πλαίσιο κάποιας αριθμολαγνείας. Το πρόγραμμά μας έπρεπε να υλοποιεί το σήμα VGA με τεράστια ακρίβεια, αφού όλες οι οθόνες --και πολύ περισσότερο οι παλιές-- είναι ιδιαίτερα αυστηρές με τους χρόνους. Όταν το σήμα αποκλίνει έστω και λίγο από το εκάστοτε πρότυπο, η οθόνη αρνείται να "κλειδώσει" και δεν δείχνει απολύτως τίποτα.



Ο παλμός HSYNC σηματοδοτεί την εκκίνηση ενός scanline (A). Αμέσως μετά ακολουθεί το back porch (B), ενώ στο τέλος του scanline βρίσκεται το front porch (D). Ανάμεσα σ' αυτά τα δύο βρίσκεται το ορατό τμήμα της γραμμής (C). Οι τρείς τάσεις περιγράφουν τα χρώματα των pixels που απαρτίζουν τη γραμμή.

Θέμα χρόνου

Η δημιουργία των παλμών HSYCN και VSYNC αποτελεί απλή υπόθεση, αφού οι συχνότητές τους (31,469KHz και 59,94Hz αντίστοιχα) είναι πολύ χαμηλές για τα δεδομένα ενός μικροελεγκτή. Όσο πρόχειρα κι αν γράφαμε τον κώδικα, είναι σίγουρο ότι δεν θα δυσκολευόμασταν καθόλου. Τα χρώματα των διαδοχικών pixels, όμως, πρέπει να μεταδίδονται με το ρυθμό που καθορίζει το pixel clock. Αυτό σημαίνει ότι τα δεδομένα της frame buffer πρέπει να εξέρχονται από αυτή με τη συχνότητα των 25,175MHz. Αν ρίξετε μια ματιά στο datasheet της μνήμης (TC551001BPL-70L), θα διαπιστώσετε ότι ο ελάχιστος χρόνος ανάγνωσης (minimum read access time) φτάνει στα 70nsec. Συνεπώς, ο μέγιστος ρυθμός ανάγνωσης προσεγγίζει μόλις τα 14,286MHz. Μήπως η επιλογή αυτής της μνήμης συνιστά σχεδιαστικό σφάλμα; Τα ψηφιακά κυκλώματα κρύβουν πολλές παγίδες σαν αυτή, αλλά να είστε σίγουροι ότι εδώ δεν έγινε λάθος. Ακόμα κι αν είχαμε χρησιμοποιήσει γρηγορότερη μνήμη, δεν θα κερδίζαμε απολύτως τίποτα. Το κύκλωμά μας περιλαμβάνει έναν ακόμα πιο αργό "παίκτη". Αναφερόμαστε στον μικροελεγκτή, που σύμφωνα με το datasheet έχει μέγιστη συχνότητα λειτουργίας τα 20MHz. Αυτό ακούγεται πολύ καλό, αλλά δεν αρκεί. Θυμηθείτε για λίγο το κύκλωμά μας. Το address bus της μνήμης καταλήγει στον μικροελεγκτή. Αυτό σημαίνει ότι για να διαβάσουμε από τη frame buffer τις τιμές μερικών pixels, ο μικροελεγκτής πρέπει να υπολογίσει και να "ζητήσει" τις κατάλληλες διευθύνσεις μνήμης. Στην καλύτερη περίπτωση, αυτό προϋποθέτει μία (!) εντολή για τον υπολογισμό της διεύθυνσης και άλλη μία για την εμφάνισή της στους ακροδέκτες. Αν ο μικροελεγκτής λειτουργεί στη μέγιστη συχνότητα των 20MHz και αν οι παραπάνω δύο εντολές εκτελούνται σε ένα κύκλο του ρολογιού (CPU cycle) η καθεμία, οι νέες διευθύνσεις μνήμης θα εμφανίζονται στους ακροδέκτες του μικροελεγκτή με συχνότητα 10MHz. Κατ' επέκταση, η μέγιστη ταχύτητα ανάγνωσης μερικών τιμών από τη frame buffer φτάνει το πολύ στα 10MHz. Εδώ συναντάμε ένα πραγματικό εμπόδιο! Το pixel clock των 25,175MHz είναι άπιαστο για το κύκλωμα γραφικών του υπολογιστή μας. Ούτε η μνήμη, ούτε και ο μικροελεγκτής θα μπορούν να εκπέμψουν τα δεδομένα για 640 pixels, μέσα στο προβλεπόμενο χρονικό διάστημα.



Η αναπαράσταση ενός frame στο επίπεδο. Η πράσινη περιοχή αποτελεί το ορατό τμήμα του frame. Οι περιοχές A και B αποτελούν τα τμήματα του horizontal video blanking, ενώ οι περιοχές C και D αντιστοιχούν στα τμήματα του vertical video blanking.

Αναγκαίος συμβιβασμός

Οι οθόνες δεν ενδιαφέρονται καθόλου για το περιεχόμενο των εικόνων που προβάλλουν. Αυτό που τις ενδιαφέρει είναι να λαμβάνουν τους παλμούς συγχρονισμού (VSYNC και HSYNC) στους σωστούς χρόνους και, επιπρόσθετα, να λαμβάνουν τα δεδομένα των ορατών pixels μέσα στα κατάλληλα χρονικά διαστήματα. Το αν θα επαναλαμβάνονται κάποια pixels ή θα είναι όλα διαφορετικά, δεν παίζει κανένα ρόλο. Πίσω απ' αυτή την τετριμμένη αλήθεια κρύβεται η λύση του προβλήματός μας. Είτε στέλνουμε δεδομένα για 640 pixels είτε για λιγότερα, η οθόνη δεν θα ενοχληθεί καθόλου. Αρκεί η εκπομπή των δεδομένων να διαρκεί ακριβώς όσο προβλέπεται από το πρότυπο. Αυτό σημαίνει ότι αν κάνουμε μια υποχώρηση στην ανάλυση της εικόνας, το πρόβλημα με το pixel clock λύνεται αυτομάτως. Ωραία, θα σκεφτείτε, αλλά πόσα pixels προλαβαίνουμε να στείλουμε στο χρονικό διάστημα που η οθόνη περιμένει 640; Όπως είδαμε, το pixel clock του δικού μας κυκλώματος μπορεί να ανέλθει το πολύ ως τα 10MHz. Με αυτό το δεδομένο, η απάντηση προκύπτει πανεύκολα:

```
Με pixel clock στα 25,175MHz εκπέμπονται 640 pixels
Με pixel clock στα 10,000MHz εκπέμπονται x pixels
```

Κάνοντας τις πράξεις, προκύπτει ότι το x είναι ίσο με 254,2. Επομένως, στο χρόνο που η οθόνη περιμένει 640 pixels, το κύκλωμα γραφικών του υπολογιστή μας μπορεί να στέλνει 254. Ξέρετε κάτι; Αυτός ο αριθμός είναι τόσο κοντά στο 256, που θα ήταν αμαρτία αν δεν κάναμε τη στρογγυλοποίηση :D Αναρωτιέστε αν θα δεχτεί κάτι τέτοιο η οθόνη; Τα 8 pixels που προηγούνται από τα 640 ορατά, όπως και τα 8 που ακολουθούν, σχηματίζουν το λεγόμενο border. Η συγκεκριμένη έννοια έχει εξαλειφθεί πλέον και η εικόνα που προβάλλουν οι σύγχρονες οθόνες εκτείνεται από άκρη σε άκρη. Ωστόσο, το σήμα VGA εξακολουθεί να προβλέπει την ύπαρξη του border και παρ' όλο που δεν χρησιμεύει για την προβολή "ωφέλιμων" δεδομένων, κανένας δεν μας εμποδίζει να το αξιοποιήσουμε για την εκπομπή δυο pixels.

Πριν προχωρήσουμε, αξίζει να κάνουμε μια μικρή ανακεφαλαίωση. Το κύκλωμα γραφικών παράγει σήμα VGA με τα τυπικά timings. Ως εκ τούτου, η εκάστοτε οθόνη "νομίζει" ότι λαμβάνει εικόνα με ανάλυση 640 x 480 pixels και με ρυθμό ανανέωσης στα 59,94Hz. Στην πραγματικότητα, μόνο το refresh rate είναι αληθινό. Η GPU παράγει εικόνα με οριζόντια ανάλυση στα 256 pixels, ενώ μια παρόμοια υποχώρηση ισχύει και για την κατακόρυφη ανάλυση. Αν το πρόγραμμά μας δημιουργούσε 480 διαφορετικές γραμμές, θα καταλήγαμε στην αλλόκοτη ανάλυση των 256 x 480 pixels. Για να αποφύγουμε αυτή την παράξενη αναλογία εικόνας και, ταυτόχρονα, να απλοποιήσουμε τη δουλειά της GPU, αποφασίσαμε να στέλνουμε κάθε γραμμή εις διπλούν. Με αυτό το κολπάκι, αν και η οθόνη εξακολουθεί να λαμβάνει 480 scanlines, η κατακόρυφη ανάλυση της εικόνας μειώνεται στο μισό. Έτσι, η κάρτα γραφικών του υπολογιστή μας προσφέρει την ανάλυση των 256 x 240 pixels. Τέλος, αν και τα πρώτα κυκλώματα VGA μπορούσαν να προβάλλουν μόνο 16 χρώματα ταυτόχρονα, το δικό μας μπορεί να προβάλλει 256! Αυτό δεν χρειάστηκε κάποια προσπάθεια από μέρους μας, αφού κάθε διεύθυνση της frame buffer αντιστοιχεί σε ένα pixel και μπορεί να αποθηκεύσει ένα ολόκληρο byte.



Commodore 64: Ο πιο επιτυχημένος υπολογιστής όλων των εποχών! Μπορεί οι πωλήσεις του να μην έχουν καμία σχέση με το θέμα μας, αλλά στην οθόνη του φαίνεται το λεγόμενο border. Πρόκειται για την περιοχή με το ανοιχτό γαλάζιο χρώμα.

Το πρόγραμμα

Έχοντας ξεδιαλύνει όλα τα θεωρητικά ζητήματα που εμπλέκονται στη δημιουργία του σήματος VGA, μπορούμε να προχωρήσουμε στον κώδικα. Όπως θα έχετε υποψιαστεί, το πρόγραμμα είναι γραμμένο σε Assembly. Η παραγωγή του σήματος VGA απαιτεί ακρίβεια σε επίπεδο CPU cycle και κάτι τέτοιο μπορεί να επιτευχθεί μόνο στην Assembly. Πριν ξεκινήσουμε την παρουσίαση, έχετε υπόψη ότι μπορείτε να κατεβάσετε τα πάντα από το http://bit.ly/dh048gpufirm. Στο πακέτο θα βρείτε τα αρχεία με τον πηγαίο κώδικα καθώς και το αρχείο HEX, που μπορεί να γραφτεί απευθείας στον μικροελεγκτή. Παρεμπιπτόντως, σημειώστε ότι εργαστήκαμε στα Windows και χρησιμοποιήσαμε τον assembler της ATMEL (avrasm2). Αν εργάζεστε στο Linux, μπορείτε να χρησιμοποιήσετε τον συμβατό assembler ονόματι avra. Ο κώδικας του προγράμματος εκτείνεται σε αρκετά διαφορετικά αρχεία. Το καθένα περιέχει συγγενικές συναρτήσεις και ο διαχωρισμός αποσκοπεί στην ανάδειξη της γενικότερης δομής του προγράμματος. Μην ξεχνάτε ότι η γλώσσα Assembly δεν είναι καθόλου φιλική και, επιπρόσθετα, μια μόνο συνάρτηση μπορεί να εκτείνεται σε δεκάδες γραμμές. Αν δεν χωρίζαμε τον κώδικα, η μελέτη και η συντήρησή του θα ήταν πρακτικά αδύνατη.

Ο κώδικας ξεκινά στο αρχείο vga-fw.asm και από εκεί φορτώνονται όλα τα υπόλοιπα, με τη βοήθεια των δηλώσεων include. Προφανώς το αρχείο 0.macros.asm περιλαμβάνει όσα macros ορίσαμε, σε μια προσπάθεια να καταστήσουμε τον κώδικα ελαφρώς πιο ευανάγνωστο. Ο κώδικας του αρχείου 1.prep.asm φροντίζει για τη λήψη των εντολών και των αντίστοιχων παραμέτρων που στέλνει ο επεξεργαστής. Μέσα στο 2.buffer.asm βρίσκονται οι ρουτίνες που χειρίζονται τη frame buffer. Για παράδειγμα, θα βρείτε τη ρουτίνα που καθαρίζει την οθόνη, τυπώνει χαρακτήρες, ρολάρει την εικόνα, κ.λπ. Οι ρουτίνες που σχετίζονται με το χειρισμό του δρομέα έχουν τοποθετηθεί στο αρχείο 2.cursor.asm. Τέλος, στα αρχεία που το όνομά τους ξεκινά με τον αριθμό 3 περιλαμβάνονται όλες οι σταθερές: Δεδομένα που δεν μεταβάλλονται κατά την εκτέλεση του προγράμματος, όπως η γραμματοσειρά, η εικόνα/λογότυπο του υπολογιστή κ.ά. Η παραπάνω περιγραφή ενδέχεται να σας προκάλεσε

σύγχυση, αφού αναφέρθηκαν τμήματα του κώδικα για τα οποία δεν έχουμε μιλήσει καθόλου. Για την ώρα μπορείτε να ξεχάσετε οτιδήποτε σας προβλημάτισε και να επικεντρώσετε στο θέμα που μας έχει απασχολήσει ως τώρα: η παραγωγή του σήματος εικόνας.

Παραγωγή γραμμών

Το πρόγραμμα καταφέρνει να μεταφέρει τα περιεχόμενα της frame buffer στην οθόνη, δημιουργώντας το ένα scanline μετά το άλλο. Όπως είδαμε νωρίτερα, οι προδιαγραφές του σήματος VGA καθορίζουν ότι τα scnalines πρέπει να εκπέμπονται με συχνότητα 31,469KHz. Αυτή η συχνότητα αντιστοιχεί σε μια περίοδο των 31,77μsec και αν λάβουμε υπόψη ότι ο μικροελεγκτής μας είναι χρονισμένος στα 20MHz, το συγκεκριμένο χρονικό διάστημα αντιστοιχεί σε περίπου 635 CPU cycles. Για να εξασφαλίσουμε τη μεγάλη ακρίβεια που απαιτεί το σήμα VGA, ο κώδικας που σχηματίζει τα scanlines τοποθετήθηκε στη ρουτίνα εξυπηρέτησης ενός interrupt (ISR). Το εν λόγω interrupt παράγεται από έναν μετρητή (timer/counter1), κάθε φορά που η τιμή του φτάνει στο 635. Τελικά, επειδή και πάλι μπορεί να σας μπερδέψαμε, κρατήστε το εξής: Όλη η δύσκολη δουλειά, δηλαδή ο σχηματισμός των διαδοχικών scnalines, πραγματοποιείται μέσα στη ρουτίνα εξυπηρέτησης ενός interrupt. Στον κώδικά μας η εν λόγω ρουτίνα έχει το (ευρηματικό) όνομα scanline και ξεκινά στη γραμμή 253 του αρχείου vga-fw.asm. Πριν προχωρήσουμε αξίζει να σημειώσουμε ότι όλες οι υπόλοιπες εργασίες της GPU (επικοινωνία με τον επεξεργαστή, εκτύπωση χαρακτήρων κ.ά.) πραγματοποιούνται κατά τη διάρκεια του vertical video blanking interval. Αυτή η περίοδος διαρκεί για 29 scanlines και οι συγκεκριμένες γραμμές περιλαμβάνουν μόνο τον παλμό HSYNC. Ως εκ τούτου, η παραγωγή τους ολοκληρώνεται πολύ γρήγορα και περισσεύει αρκετός χρόνος μέχρι το επόμενο interrupt.

Λίγο πίσω από την ακτίνα

Μη νομίζετε ότι το σήμα VGA έχει το μονοπώλιο στην αυστηρότητα ή στην πολυπλοκότητα. Τα αναλογικά τηλεοπτικά σήματα παρουσιάζουν τις ίδιες ιδιοτροπίες και, επιπρόσθετα, εκτός από εικόνα μεταφέρουν και ήχο. Αν πιστεύετε ότι η σύνθεση σήματος VGA αποτελεί την πιο επίπονη δουλειά για έναν προγραμματιστή, πρέπει να νιώθετε και τυχεροί που δεν γεννηθήκατε μερικές δεκαετίες νωρίτερα. Στα μέσα της δεκαετίας του 80, η έννοια της παιχνιδομηχανής ήταν σχεδόν ταυτισμένη με τα συστήματα της Atari. Δεν αναφερόμαστε στους ομώνυμους υπολογιστές, αλλά στις συσκευές που συνδέονταν στην τηλεόραση και είχαν σαν μοναδική αποστολή το παιχνίδι. Ένας λόγος που οδήγησε στην επιτυχία τους ήταν το πολύ χαμηλό κόστος. Ξέρετε πώς το εξασφάλισαν αυτό οι μηχανικοί της εταιρείας; Εκείνη την εποχή οι μνήμες αποτελούσαν μακράν το πιο ακριβό εξάρτημα σε ένα σύστημα. Οι παιχνιδομηχανές Atari, λοιπόν, στηρίζονταν σ' ένα κύκλωμα γραφικών (το περίφημο TIA) που δεν διέθετε μνήμη για την αποθήκευση της εικόνας. Εν ολίγοις, τα Atari δεν διέθεταν frame buffer! Σε αυτά τα συστήματα η παραγωγή του τηλεοπτικού σήματος αποτελούσε ευθύνη του hardware, αλλά αυτό δεν διευκόλυνε καθόλου τη ζωή των προγραμματιστών. Τα προγράμματα

έπρεπε να υπολογίζουν τη θέση των διαφόρων αντικειμένων και να σχεδιάζουν όλα τα γραφικά ξανά και ξανά. Μάλιστα, κάθε γραμμή της εικόνας έπρεπε να σχεδιάζεται ακριβώς τη στιγμή που θα ξεκινούσε η εκπομπή του αντίστοιχου scanline. Γι' αυτό το λόγο, οι προγραμματιστικές τεχνικές που αναπτύχθηκαν στα Atari έμειναν στην ιστορία με το γενικό όνομα "racing the beam". Τι έχετε να πείτε για τους προγραμματιστές που έγραφαν τα παιχνίδια της πλατφόρμας;



Η πιο διάσημη παιχνιδομηχανή του 80! Οι gamers την αγαπούσαν για το χαμηλό κόστος και τη μεγάλη ποικιλία παιχνιδιών, ενώ οι προγραμματιστές την αντιπαθούσαν εξαιτίας του πολύ περίπλοκου και δύσκολου προγραμματισμού της. Παρεμπιπτόντως, σε αυτή την παιχνιδομηχανή έχουμε παίξει Berzerk, Dig Dug, Defender, Combat και Outlaw! Εσείς;

Περισσότερη ακρίβεια

Η ανάγκη να ικανοποιήσουμε τόσο αυστηρούς χρονικούς περιορισμός προκύπτει πολύ σπάνια. Ως αποτέλεσμα, δεν ασχολούμαστε ποτέ με το πόσος χρόνος μεσολαβεί από την εμφάνιση ενός interrupt έως την εκτέλεση της αντίστοιχης ρουτίνας χειρισμού. Για τη συντριπτική πλειονότητα των εφαρμογών, αυτό το χρονικό διάστημα μπορεί να θεωρηθεί μηδενικό. Στην πραγματικότητα, τα πράγματα δεν είναι τόσο απλά (κλασικά). Το εν λόγω χρονικό διάστημα δεν είναι ούτε μηδενικό, ούτε σταθερό! Όταν προκύπτει κάποιο interrupt ο μικροελεγκτής μεταβαίνει σε μια ειδική διεύθυνση της μνήμης προγράμματος, που ονομάζεται interrupt vector. Φυσικά, για κάθε interrupt υπάρχει κι ένα αντίστοιχο interrupt vector. Μπορείτε να δείτε τις σχετικές διευθύνσεις στην ενότητα "Interrupt Vectors in ATmega644" του datasheet. Εξαιτίας των παραπάνω, το πρόγραμμά μας ξεκινά κάπως έτσι:

```
.org 0
     rjmp mcu_init        ; Reset-Interrupt Handler
.org OC1Aaddr
     rjmp scanline        ; Timer1 CompareA-Interrupt Handler
```

Στη διεύθυνση που έχει αντιστοιχιστεί το reset interrupt, έχουμε τοποθετήσει μια παραπομπή προς το σημείο εκκίνησης του προγράμματός μας -- τον κώδικα που αρχικοποιεί τον μικροελεγκτή. Στη διεύθυνση OC1Aaddr, που αντιστοιχεί στο interrupt που αξιοποιεί το πρόγραμμά μας, έχουμε τοποθετήσει μια παραπομπή προς την αντίστοιχη ρουτίνα εξυπηρέτησης. Πιστεύουμε ότι όλα αυτά είναι απλά και κατανοητά. Δείτε όμως μια ύπουλη λεπτομέρεια: Όταν προκύπτει κάποιο interrupt, ο μικρο-

ελεγκτής δεν διακόπτει την εκτέλεση της τρέχουσας εντολής. Περιμένει την ολοκλήρωσή της και μόνο τότε μεταβαίνει στην κατάλληλη διεύθυνση (interrupt vector). Αυτό σημαίνει ότι, αν εκείνη τη στιγμή εκτελείται μια εντολή που διαρκεί παραπάνω από ένα CPU cycle, η εκτέλεση της ρουτίνας εξυπηρέτησης θα καθυστερήσει κατά δύο ή περισσότερα CPU cycles. Τελικά, από την εμφάνιση ενός interrupt μέχρι την εκτέλεση του αντίστοιχου ISR, μεσολαβούν από 5 έως 9 CPU cycles. Ε, λοιπόν, αυτή η αβεβαιότητα συνιστά μεγάλο πρόβλημα για το πρόγραμμά μας. Αν τα διάφορα scanlines δημιουργούνται άλλοτε νωρίτερα και άλλοτε αργότερα, η συχνότητα παραγωγής τους δεν θα είναι σταθερή. Κατά συνέπεια, η οθόνη θα δυσκολεύεται να "κλειδώσει" στο σήμα και θα παρατηρούμε ένα τρεμόπαιγμα ή δεν θα βλέπουμε τίποτα.

Για καλή μας τύχη, οι εντολές που διαρκούν 3, 4 ή 5 CPU cycles είναι πολύ λίγες και στο δικό μας πρόγραμμα έχουν χρησιμοποιηθεί ελάχιστα. (Η αλήθεια είναι ότι δεν καταβάλαμε μεγάλη προσπάθεια, αφού οι εν λόγω εντολές σχετίζονται με ελέγχους και λειτουργίες που χρησιμοποιούνται σπάνια.) Αυτό σημαίνει ότι κάθε φορά που προκύπτει το interrupt για την παραγωγή μιας γραμμής, εισάγεται καθυστέρηση των 5 ή των 6 CPU cycles. Αυτή η ασάφεια εξακολουθεί να αποτελεί πρόβλημα, αλλά έχει μικρότερο εύρος και αντιμετωπίζεται πιο εύκολα. Δείτε πώς ξεκινάει η ρουτίνα που δημιουργεί τα scanlines:

```
scanline:
     lds isr_temp1, TCNT1L    ; (2)
     sbrc isr_temp1, 0        ; (1~2)
     rjmp vdelay_end          ; (2)
vdelay_end:
```

Αναρωτιέστε τι πετυχαίνουμε εδώ; Όταν ο χρόνος για την εκκίνηση της ρουτίνας φτάνει στους 5 κύκλους (CPU cycles), οι παραπάνω γραμμές εισάγουν μια καθυστέρηση που διαρκεί επίσης 5 κύκλους. Όταν όμως το αρχικό χρονικό διάστημα φτάνει στους 6 κύκλους, ο κώδικας εισάγει μια καθυστέρηση των 4 κύκλων. Έτσι, η παραγωγή των scanlines ξεκινάει πάντα με την ίδια καθυστέρηση κι επαναλαμβάνεται με απόλυτα σταθερή συχνότητα! Ας δούμε πώς το πετύχαμε αυτό. Όπως είπαμε, το interrupt για τη δημιουργία των γραμμών προκύπτει κάθε φορά που ο timer/counter1 φτάνει στην τιμή 635. Ωστόσο, την ίδια στιγμή ο μετρητής μηδενίζεται και αρχίζει να αυξάνεται από το επόμενο CPU cycle, έως ότου φτάσει και πάλι στο 635. Αν η καθυστέρηση που θέλουμε να αντισταθμίσουμε φτάσει στους 5 κύκλους, όταν ξεκινήσει η ρουτίνα scanline ο μετρητής θα έχει την τιμή 5. Αντίστοιχα, αν η καθυστέρηση ανέλθει στους 6 κύκλους, ο μετρητής θα έχει την τιμή 6. Όπως βλέπετε, στην πρώτη περίπτωση η τιμή του μετρητή είναι περιττή και στη δεύτερη είναι άρτια. Θυμηθείτε το αυτό. Η πρώτη εντολή του παραπάνω αποσπάσματος λαμβάνει το "λιγότερο σημαντικό" byte (Least Significant Byte) της τιμής του μετρητή και το αποθηκεύει στον καταχωριστή isr_temp1. Στη συνέχεια ελέγχουμε αν το πρώτο bit του συγκεκριμένου καταχωριστή είναι 0 ή 1. Στην πρώτη περίπτωση (άρτια τιμή) η εντολή ελέγχου διαρκεί δύο κύκλους και η επόμενη εντολή παρακάμπτεται.

Στη δεύτερη περίπτωση (περιττή τιμή) η εντολή ελέγχου διαρκεί έναν κύκλο και η επόμενη εντολή, που διαρκεί για δύο κύκλους, εκτελείται κανονικά. Αν κάνετε τις προσθέσεις, θα διαπιστώσετε ότι σε κάθε περίπτωση το άθροισμα των καθυστερήσεων ισούται με 10 κύκλους. Απλό δεν ήταν;

Που είμαι; Τι κάνω;

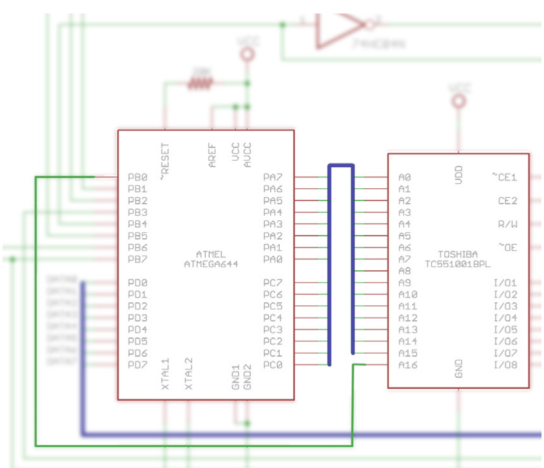
Ελπίζουμε να μη σας τρόμαξε η προηγούμενη παράγραφος. Πάντως το απόσπασμα που εξετάσαμε αποτελεί το πιο σκοτεινό τμήμα ολόκληρου του προγράμματος. Ο υπόλοιπος κώδικας είναι συγκριτικά απλούστερος. Η ρουτίνα scanline δημιουργεί τον παλμό οριζόντιου συγχρονισμού και ταυτόχρονα πραγματοποιεί μερικές άσχετες εργασίες. Σε αυτές θα αναφερθούμε στο επόμενο άρθρο της σειράς, όταν θα εξετάσουμε τις υπόλοιπες λειτουργίες του προγράμματος. Μετά από τον παλμό HSYNC ακολουθεί το back porch, ενώ στη συνέχεια πραγματοποιείται η ανάγνωση μιας γραμμής από τη frame buffer. Τα σχετικά δεδομένα αποτελούν το "ορατό" τμήμα του scanline και γι' αυτό κατευθύνονται προς τον DAC κι από 'κεί προς την οθόνη. Η ρουτίνα scanline δεν κάνει τίποτα περίπλοκο, αλλά είναι βέβαιο ότι η περιγραφή μάς έχει γεννήσει νέα ερωτήματα: Πότε δημιουργείται ο κατακόρυφος παλμός συγχρονισμού (VSYNC); Ποιες γραμμές είναι ενεργές και ποιες ανήκουν στο vertical video blanking interval; Σε ποια διεύθυνση της frame buffer βρίσκονται τα δεδομένα της εκάστοτε γραμμής;

Ο κώδικας θα μπορούσε να διατηρεί έναν μετρητή γραμμών και, σύμφωνα με την τιμή του, να υπολογίζει την απάντηση σε όλα τα παραπάνω ερωτήματα. Ωστόσο ένας τέτοιος μηχανισμός θα ήταν αρκετά χρονοβόρος και δεν θα άφηνε κανένα περιθώριο για τις "άσχετες" εργασίες που αναφέραμε προηγουμένως. Αυτό μας υποχρέωσε να υιοθετήσουμε μια λύση που ήταν λίγο πιο κουραστική για εμάς, αλλά πολύ πιο ξεκούραστη για το πρόγραμμα. Συγκροτήσαμε έναν πίνακα με ένα ζεύγος τιμών για κάθε μία από τις 525 γραμμές του frame. Η πρώτη τιμή του ζευγαριού (το πρώτο byte) περιλαμβάνει μια σημαία (flag) που δείχνει αν πρέπει να ενεργοποιηθεί ο παλμός κατακόρυφου συγχρονισμού ή όχι. Το ίδιο byte περιλαμβάνει και ένα δεύτερο flag, που δείχνει αν η συγκεκριμένη γραμμή ανήκει στην περιοχή του vertical video blanking. Η δεύτερη τιμή σε κάθε ζευγάρι του πίνακα χρησιμεύει για τις ορατές γραμμές. Αποτελεί το "περισσότερο σημαντικό" byte (Most Significant Byte) της διεύθυνσης μέσα στη frame buffer, όπου βρίσκονται οι τιμές των pixels της τρέχουσας γραμμής. Όπως αντιλαμβάνεστε, η συγκρότηση αυτού του πίνακα ήθελε μεγάλη προσοχή και υπομονή. Όταν ολοκληρώθηκε όμως δεν χρειάστηκε να επαναληφθεί ποτέ ξανά, ενώ το πρόγραμμα είχε αποκτήσει έναν γρήγορο τρόπο για να γνωρίζει σε ποια γραμμή βρίσκεται και τι πρέπει να κάνει. Μπορείτε να δείτε τον πίνακα στο αρχείο 3.lines.asm. Η ρουτίνα scanline χρησιμοποιεί έναν δείκτη για να διαβάζει κάθε φορά ένα ζεύγος τιμών από τον πίνακα. Ο συγκεκριμένος δείκτης αυξάνεται αυτόματα μετά από κάθε ανάγνωση κι έτσι είναι πάντοτε έτοιμος προς χρήση. Προσέξτε τώρα μια ενδιαφέρουσα λεπτομέρεια: Στο τέλος του πίνακα έχουμε προσθέσει ένα byte με την τιμή 255, η οποία δεν εμφανίζεται σε κανένα άλλο στοιχείο του πίνακα. Ο κώδικας τσεκάρει το byte που λαμβάνει κάθε φορά και αν διαπιστώσει ότι έχει την τιμή 255 αντιλαμβάνεται ότι έχει φτάσει στο τέλος του πί-

νακα κι επαναφέρει τον δείκτη στην αρχική του τιμή. Με αυτόν τον τρόπο, ο δείκτης βρίσκεται πάντοτε μέσα σε αποδεκτά όρια και η ρουτίνα scanline δεν χρειάζεται να πραγματοποιεί χρονοβόρους ελέγχους, ούτε να θυμάται τον αριθμό της τρέχουσας γραμμής. Ωραίο κολπάκι, δεν μπορείτε να πείτε ;)

Ορατά pixels

Μέχρι στιγμής έχουμε αναφέρει επανειλημμένα ότι τα δεδομένα της εικόνας αποθηκεύονται στη frame buffer, αλλά δεν έχουμε πει τίποτα για την οργάνωσή τους. Κατ' αρχάς, σημειώστε ότι εφόσον η εικόνα έχει ανάλυση 256 x 240 και χρησιμοποιούμε ένα byte για κάθε pixel, η αποθήκευση μιας ολόκληρης εικόνας απαιτεί 61440 bytes. Ας ξεκινήσουμε τώρα από το τσιπάκι μνήμης και τον τρόπο που το χειριζόμαστε. Επειδή η χωρητικότητά του είναι 128KB, διαθέτει 17 εισόδους διεύθυνσης (2^17 = 128K). Ωστόσο, αν ρίξετε μια ματιά στο κύκλωμα θα διαπιστώσετε ότι το address bus που συνδέει τη μνήμη με τον μικροελεγκτή έχει εύρος 16 bits. Η δέκατη έβδομη είσοδος διεύθυνσης (το περισσότερο σημαντικό ψηφίο) συνδέεται σε έναν άσχετο ακροδέκτη του μικροελεγκτή. Από αυτόν, ο μικροελεγκτής μπορεί να επιλέγει το αν θα εργάζεται στα "άνω" 64KB ή στα "κάτω". Εν ολίγοις, μέσα στο τσιπάκι μνήμης υπάρχει χώρος για τη φιλοξενία δύο frame buffers, ενώ υπάρχει και η υποδομή για τον χειρισμό τους. Πάντως πρέπει να ομολογήσουμε ότι ο κώδικας που έχουμε γράψει μέχρι στιγμής αξιοποιεί μόνο τη μία buffer.



Σε αυτή την περιοχή του κυκλώματος φαίνεται το address bus, που συνδέει τη μνήμη με τον μικροελεγκτή. Παρατηρήστε ότι το "περισσότερο σημαντικό" bit των διευθύνσεων που δέχεται η μνήμη (είσοδος A16) συνδέεται στον ακροδέκτη PB0 του μικροελεγκτή και δεν ανήκει στο address bus. Από αυτό το bit, ο μικροελεγκτής μπορεί να καθορίζει αν το address bus θα παραπέμπει σε κάποια θέση των "άνω" 64KB ή των "κάτω".

Ας εστιάσουμε και στο εσωτερικό των buffers. Το μέγεθος της κάθε μίας είναι 64KB, ενώ μια ολόκληρη εικόνα απαιτεί μόλις 61440 bytes. Αυτά τα δεδομένα φροντίζουμε να αποθηκεύονται με τη σειρά που εμφανίζονται και στην οθόνη. Έτσι, τα 256 bytes που περιγράφουν τα pixels της πρώτης γραμμής της εικόνας, αποθηκεύονται στις πρώτες 256 θέσεις της μνήμης και πάει λέγοντας. Με αυτόν τον τρόπο κι επειδή κάθε γραμμή περιγράφεται με ακριβώς 256 bytes, προκύπτει ένα πολύ βολικό σύστημα διευθυνσιοδότησης: Το "περισσότερο σημαντικό" byte του address bus ανα-

φέρεται στη γραμμή της εικόνας, ενώ το "λιγότερο σημαντικό" byte αναφέρεται στο pixel της επιλεγμένης γραμμής. Για παράδειγμα, αν θέλουμε να διαβάσουμε την τιμή του 23ου pixel της 100ης γραμμής της εικόνας, αρκεί να τοποθετήσουμε στο high-byte του address bus την τιμή 100 και στο low-byte την τιμή 23. Αυτό το σύστημα είναι απλό και κατανοητό από τους ανθρώπους, ενώ επιτρέπει και την ταχύτατη πρόσβαση στη frame buffer.

Θυμόσαστε τη συζήτηση για το μέγιστο pixel clock που μπορεί να πετύχει το κύκλωμά μας; Για να "πιάσουμε" την ταχύτητα των 10MHz, οι διευθύνσεις μνήμης πρέπει να παράγονται με μία εντολή του ενός CPU cycle! Ε, λοιπόν, με τον τρόπο που έχουμε οργανώσει τα δεδομένα μέσα στο frame buffer, κάτι τέτοιο είναι απόλυτα εφικτό. Λίγο πριν ξεκινήσει το ορατό τμήμα του εκάστοτε scanline (όσο βρισκόμαστε στο back porch), φορτώνουμε στο high-byte του address bus τον αριθμό της αντίστοιχης γραμμής. Προφανώς, εφόσον "εκπέμπουμε" ένα συγκεκριμένο scanline, αυτό το μέγεθος δεν πρόκειται να αλλάξει και μπορούμε να το ξεχάσουμε. Επιπρόσθετα, μέσα στο χρόνο του back porch θέτουμε την τιμή ενός προσωρινού καταχωριστή σε 255. Αυτός ο καταχωριστής χρησιμοποιείται σαν το low-byte της διεύθυνσης. Όταν ξεκινήσει το ορατό τμήμα της γραμμής, θέλουμε να πιάσουμε τη μέγιστη ταχύτητα υπολογισμού και εξόδου των διευθύνσεων. Έτσι, το πρόγραμμά μας επαναλαμβάνει μόνο τις εξής δύο ενέργειες: Αυξάνει τον καταχωριστή με το low-byte κατά μία μονάδα και μεταφέρει την τιμή του στους ακροδέκτες. Καθεμία από τις συγκεκριμένες εντολές διαρκεί ακριβώς ένα CPU cycle και αυτό το ζεύγος επαναλαμβάνεται 256 φορές. Την πρώτη φορά που επαναλαμβάνεται το ζεύγος ο καταχωριστής με το low-byte μηδενίζεται (η πρόσθεση μιας μονάδας σε ένα byte που έχει την τιμή 255 έχει ως συνέπεια την υπερχείλιση του byte και την "επιστροφή" του στο μηδέν), ενώ την τελευταία φορά φτάνει στην τιμή 255. Οι 256 επαναλήψεις των δύο εντολών βρίσκονται αραδιασμένες σε ένα macro με το όνομα get_line_data, στο αρχείο 0.macros.asm. Με αυτή την άκομψη αλλά ταχύτατη μέθοδο, πετυχαίνουμε το στόχο μας κι ανεβάζουμε το pixel clock στα 10MHz.

Κάπου εδώ θα σας αφήσουμε να μελετήσετε τον κώδικα της ρουτίνας scanline. Δεδομένου ότι είναι γραμμένος σε Assembly, μην προσπαθείτε να ερμηνεύσετε κάθε γραμμή ξεχωριστά. Ακόμα κι ένας απλός έλεγχος που σε άλλες γλώσσες θα ολοκληρωνόταν σε μια γραμμή, στην Assembly ενδέχεται να εκτείνεται σε τρεις ή σε τέσσερις. Είναι προτιμότερο να σταθείτε στη γενική λειτουργία του κώδικα και να μη μπλέξετε με τις ιδιαιτερότητες της γλώσσας. Εξάλλου, η εκμάθηση της Assembly δεν θα μπορούσε να ξεκινήσει μ' ένα τόσο περίπλοκο πρόγραμμα. Στο επόμενο άρθρο της σειράς θα εξετάσουμε τα υπόλοιπα τμήματα του κώδικα και το παζλ θα συμπληρωθεί: θα δείτε πώς φτάσαμε από την παραγωγή των scanlines σε μια αργόστροφη αλλά λειτουργική GPU!

Skill: Intermediate
Tags: .NET, C#, managed code, unmanaged code

Ανάπτυξη εφαρμογών σε C# για το .NET

[μέρος 10ο]

Σ' αυτή τη σειρά άρθρων ξεκινήσαμε δειλά δειλά από τα πιο βασικά στοιχεία της C# και φτάσαμε να δημιουργούμε εφαρμογές για το περιβάλλον γραφικών των Windows. Στο παρόν άρθρο θα επεκταθούμε σε ελαφρώς πιο προχωρημένα χαρακτηριστικά αυτής της πανίσχυρης γλώσσας και θα σας αφήσουμε να συνεχίσετε το ταξίδι μόνοι σας.

του Πέτρου Κυλαδίτη

Στο προηγούμενο τεύχος αναβαθμίσαμε τις λειτουργίες του deltaPad, ενός απλού text editor που θα θέλαμε να αντικαταστήσει πλήρως το Notepad των Windows. Ο στόχος επετεύχθη αρκετά εύκολα και προχωρήσαμε παραπέρα, εξετάζοντας και κάποιες βελτιώσεις. Στο παρόν άρθρο θα ασχοληθούμε και πάλι με το deltaPad, αλλά δεν θα περιοριστούμε σ' αυτό. Θα ξεκινήσουμε με μια ενδιαφέρουσα προσθήκη στις δυνατότητες του μικρού μας editor και στη συνέχεια θα περάσουμε σε κάποια άσχετα θέματα, που αναδεικνύουν την ευελιξία της C#.

Όλα τα παραθυρικά περιβάλλοντα, από την εποχή που έκαναν την πρώτη τους εμφάνιση στο Xerox PARC (1967), διέθεταν τη λειτουργία του drag'n'drop. Οι προγραμματιστές είχαν σκεφτεί από τότε την ευκολία που θα προσέφερε η ικανότητα να "σέρνεις" κάτι και να το "αποθέτεις" πάνω σε κάτι άλλο, χρησιμοποιώντας το ποντίκι ή άλλες παρεμφερείς συσκευές εισόδου, όπως τα touch tablets (οι πρόγονοι των γνωστών touchpads). Το drag'n'drop βέβαια δεν χρειάζεται συστάσεις και είναι περισσότερο από ότι το έχετε ήδη χρησιμοποιήσει (π.χ., για να "ρίξετε" ένα ή περισσότερα αρχεία σε κάποιο πρόγραμμα επεξεργασίας, για τη μεταφορά κειμένου από ένα σημείο σε κάποιο άλλο κ.ο.κ.). Η τελευταία προσθήκη που θα κάνουμε στο deltaPad, αφορά ακριβώς στην υποστήριξη του drag'n'drop.



Ένας υπολογιστής Xerox Alto (1973), εφοδιασμένος με το πρώτο παραθυρικό περιβάλλον εργασίας. Δίπλα του φαίνεται ένα ποντίκι (δεξιά) και ένα touch tablet (αριστερά), ο πρόγονος των touchpads.

Drag'n'drop στο deltaPad

Αρκετοί από εσάς μπορεί να το έχετε διαπιστώσει ήδη: Το deltaPad δεν επιτρέπει τη μεταφορά και την απόθεση κειμένου, ενώ αν του "πετάξουμε" ένα αρχείο κειμένου δεν θα κάνει τίποτα. Δεν θα πρέπει να σας παραξενεύει αυτό. Το drag'n'drop δεν αποτελεί αποκλειστική υπόθεση του λειτουργικού συστήματος. Φυσικά, η πράξη της απόθεσης εντοπίζεται από το σύστημα, το οποίο σπεύδει να ενημερώσει την εκάστοτε εφαρμογή. Ωστόσο ο χειρισμός αυτών των συμβάντων αποτελεί ευθύνη της εκάστοτε εφαρμογής. Με άλλα λόγια, αν μια εφαρμογή δεν διαθέτει την κατάλληλη πρόβλεψη, ό,τι κι αν κάνει ο χρήστης με το ποντίκι, όσο έξυπνο κι αν είναι το λειτουργικό σύστημα, δεν θα συμβεί τίποτα. Για τη λειτουργία του drag'n'drop απαιτείται η παρουσία των κατάλληλων event handlers. Ο πίνακας που ακολουθεί περιλαμβάνει τα κυριότερα events που παράγονται από τα Windows και σχετίζονται με τη λειτουργία drag'n'drop...

DragEnter	Λαμβάνει χώρα μόλις ο χρήστης σύρει κάποιο στοιχείο σε περιοχή που είναι ενεργοποιημένη η λειτουργία drag'n'drop.
DragOver	Πυροδοτείται κατ' επανάληψη καθ' όλη τη διάρκεια που ο χρήστης μετακινεί με το ποντίκι του κάτι πάνω από μια περιοχή drag'n'drop.
DragLeave	Πυροδοτείται όταν το ποντίκι εξέλθει από την επιφάνεια που δέχεται το drag'n'drop.
DragDrop	Το βασικότερο συμβάν που πυροδοτείται μόλις ο χρήσης απελευθερώσει το κουμπί του ποντικιού του, όταν δηλαδή αποθέσει το στοιχείο που σύρει κατά τη διάρκεια του drag'n'drop.

Στο deltaPad θα αξιοποιήσουμε το πρώτο και το τέταρτο event. Με το πρώτο θα δημιουργούμε ένα οπτικό αποτέλεσμα που θα ενημερώνει το χρήστη ότι το drag'n'drop λειτουργεί, ενώ με το δεύτερο θα πραγματοποιείται ο χειρισμός των στοιχείων που απέθεσε ο χρήστης. Αν πρόκειται για κείμενο, θα εισάγεται στο σημείο όπου βρίσκεται ο κέρσορας. Αν πρόκειται για κάποιο αρχείο, θα εκτελείται η διαδικασία ανοίγματος και το περιεχόμενό του θα εμφανίζεται στο παράθυρο του editor.



Το πληκτρολόγιο του υπολογιστή Xerox Star (1981) έφερε ειδικά κουμπιά για τις διάφορες χρήσεις του ποντικιού στο πρώιμο περιβάλλον γραφικών του. Ανάμεσα σε άλλα, το drag'n'drop πραγματοποιούνταν με τη βοήθεια του κουμπιού move.

Το drag'n'drop θέλουμε να λειτουργεί επάνω στο textbox, εφόσον εκεί εμφανίζεται όποιο αρχείο έχουμε ανοίξει. Εξάλλου, το textbox καταλαμβάνει το μεγαλύτερο μέρος του παραθύρου του προγράμματος κι αποτελεί τον πιο εύκολο "στόχο" για την απόθεση αρχείων. Για να δημιουργήσουμε τους χειριστές για τα δύο events που αναφέραμε παραπάνω, επιλέγουμε το textbox και μεταβαίνουμε στο παράθυρο "Ιδιότητες", από όπου και θέτουμε την ιδιότητα AllowDrop σε true. Κατόπιν μεταβαίνουμε στην καρτέλα "Συμβάντα" του ίδιου παραθύρου και προσθέτουμε τις μεθόδους με τον γνωστό τρόπο. Ας δούμε τώρα τον κώδικα που περιλαμβάνει η μέθοδος χειρισμού του DragEnter:

```
if (e.Data.GetDataPresent(DataFormats.FileDrop) ||
      e.Data.GetDataPresent(DataFormats.Text)) {
         e.Effect = DragDropEffects.Copy ;
}
```

Η παράμετρος e, που παρέχεται αυτόματα στη μέθοδο χειρισμού του event, αποτελεί αντικείμενο της κλάσης DragEventArgs. Με τον έλεγχο if τσεκάρουμε αν τα δεδομένα που έχουν "συρθεί" είναι τύπου FileDrop ή Text. Με άλλα λόγια, τσεκάρουμε αν πρόκειται για κάποιο αρχείο ή για κείμενο. Σε αυτές τις δύο περιπτώσεις αποδίδουμε στην ιδιότητα Effect την τιμή Copy, της απαρίθμησης DragDropEffects. Αυτό έχει το εξής αποτέλεσμα: Τη στιγμή που ο χρήστης σέρνει με το ποντίκι του κάποιο αρχείο ή κάποιο τμήμα κειμένου, ο δείκτης του ποντικού αλλάζει και προστίθεται στην όψη του ένα "+". Σε ό,τι αφορά στο δικό μας πρόγραμμα (τον παραλήπτη των δεδομένων) το "effect" δεν επιδρά και στη συγκεκριμένη περίπτωση (Copy) δεν πετυχαίνει τίποτα, πέρα από το να ενημερώνει το χρήστη ότι το drag'n'drop λειτουργεί. Αντίθετα, σε ό,τι αφορά στην προέλευση των δεδομένων το "effect" παίζει σημαντικό ρόλο. Αν επιλέγαμε το Move αντί του Copy κι αν ο χρήστης είχε σύρει κάποιο απόσπασμα κειμένου, κατά την απόθεση (drop) το κείμενο θα διαγραφόταν από την αρχική του θέση. Με τη χρήση του Copy, όμως, η πηγή των δεδομένων δεν επηρεάζεται καθόλου. Η απαρίθμηση DragDropEffects περιλαμβάνει τα εξής μέλη:

- `DragDropEffects.Move`
- `DragDropEffects.Copy`
- `DragDropEffects.Link`
- `DragDropEffects.Scroll`
- `DragDropEffects.All`
- `DragDropEffects.None`

Το τελευταίο έχει ως αποτέλεσμα την εμφάνιση του συμβόλου της απαγόρευσης και αξιοποιείται για να δηλώσει ότι το drag'n'drop δεν υποστηρίζεται ή δεν επιτρέπεται. Η λειτουργία των υπολοίπων είναι λίγο πολύ προφανής.

Όποιο effect κι αν επιλέξουμε, αν δεν γράψουμε κώδικα για το χειρισμό των εισερχόμενων δεδομένων δεν θα συμβεί τίποτα. Αυτό μπορούμε να το διαπιστώσουμε εύκολα, κάνοντας μια μεταγλώττιση και μερικές δοκιμές με το ποντίκι. Για να λειτουργήσει το drag'n'drop σύμφωνα με τις προσδοκίες μας πρέπει να τις υλοποιήσουμε (τις προσδοκίες :D). Ας δούμε τον κώδικα που γράψαμε για το event DragDrop:

```
void TextBoxDragDrop(object sender, DragEventArgs e)
{
    if(e.Data.GetDataPresent(DataFormats.FileDrop)) {
        string[] droppedFiles =
                 (string[])e.Data.GetData(DataFormats.FileDrop);
        LoadText(droppedFiles[0]);
    }
    if (e.Data.GetDataPresent(DataFormats.Text)){
        string droppedText = (string)e.Data.GetData(DataFormats.Text);
        int cursorPosition = textBox.SelectionStart;
        textBox.Text = textBox.Text.Insert(cursorPosition, droppedText);
        textBox.SelectionStart = cursorPosition + droppedText.Length;
    }
}
```

Όπως μπορείτε να διαπιστώσετε εύκολα, εξετάζουμε κι αντιμετωπίζουμε κάθε περίπτωση διαφορετικά. Ο βασικός έλεγχος, ο τύπος των δεδομένων που "έριξε" ο χρήστης, πραγματοποιείται όπως και πριν.

Στην περίπτωση της απόθεσης αρχείων χρησιμοποιούμε τη μέθοδο GetData, ώστε να διαβάσουμε τις πληροφορίες που μεταφέρει το αντικείμενο DragEventArgs. Σε αυτή τη μέθοδο δίνουμε ως παράμετρο το FileDrop κι έτσι λαμβάνουμε έναν πίνακα με τα ονόματα των αρχείων που απόθεσε ο χρήστης. Επειδή το deltaPad μπορεί να ανοίγει μόνο ένα κείμενο τη φορά, στην περίπτωση που ο χρήστης έχει αποθέσει πολλά, το πρόγραμμα ασχολείται μόνο με το πρώτο. Ακριβώς γι' αυτό, διαβάζουμε το στοιχείο του πίνακα υπ αριθμόν 0. Αυτό το στοιχείο χρησιμοποιείται ως παράμετρος για την κλήση της LoadText που, όπως μπορείτε να δείτε, έχει υποστεί ανανέωση:

```
private void LoadText(string filePath) {
    if(textChanged) {
        if(AskSave() == DialogResult.Cancel)
            return;
    }

    if(filePath == null) {
```

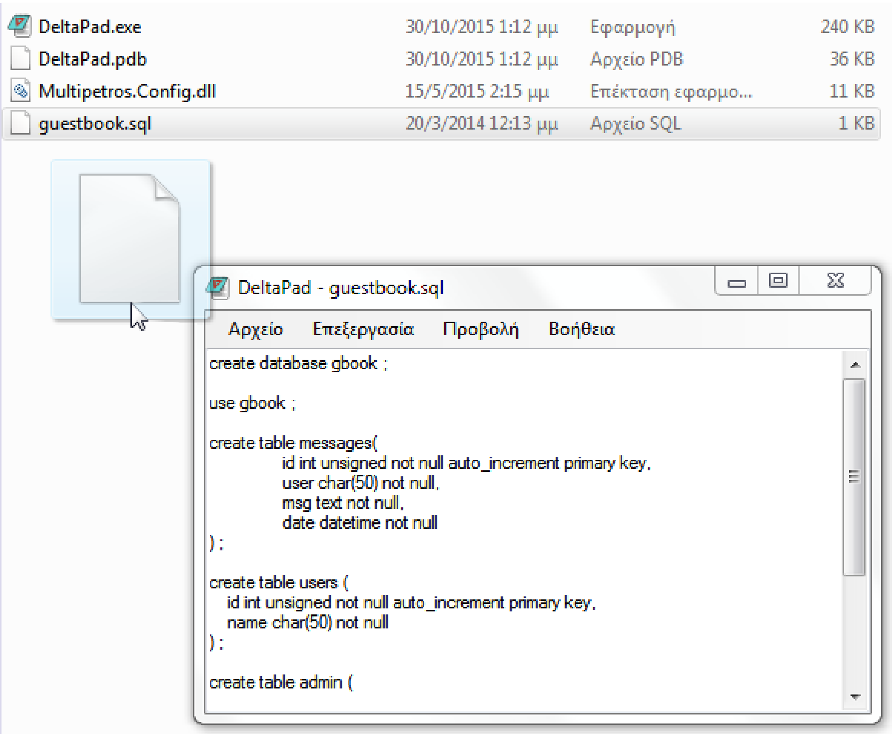
```
        DialogResult result = openFileDialog.ShowDialog(this);
        if(result == DialogResult.OK) {
            filePath = openFileDialog.FileName;
        } else {
            return;
        }
    }

    try {
        textBox.Text = File.ReadAllText(filePath) ;
        textChanged = false;
        this.Text = "DeltaPad - " + Path.GetFileName(filePath);
    } catch(Exception err) {
        MessageBox.Show(err.Message, "Σφάλμα φόρτωσης αρχείου",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Όπως βλέπετε, η ανανεωμένη LoadText δέχεται την παράμετρο filePath. Προφανώς, πρόκειται για μια συμβολοσειρά που περιέχει το όνομα του προς φόρτωση αρχείου. Ο κώδικας της μεθόδου τσεκάρει αν η παράμετρος έχει τιμή ή όχι (null) και στη δεύτερη περίπτωση εμφανίζει το παράθυρο επιλογής αρχείου. Έτσι, όταν η LoadText καλείται από κάποιο συμβάν drag'n'drop, η παράμετρος filePath διαθέτει τιμή και το πρόγραμμα φορτώνει το αρχείο χωρίς να εμφανίζει το παράθυρο επιλογής. Η επέμβαση που κάναμε στη loadText, ώστε να περιμένει πλέον μία παράμετρο, θα δημιουργήσει προβλήματα στα σημεία όπου γινόταν κλήση της μεθόδου χωρίς παράμετρο. Επομένως, πρέπει να επισκεφθούμε κάθε σημείο από το οποίο καλείται η LoadText και να προσθέσουμε την παράμετρο null. Για παράδειγμα, όταν κάνουμε κλικ επάνω στην επιλογή Open του μενού, η κλήση της LoadText πρέπει να γίνεται ως εξής:

```
void OpenToolStripMenuItemClick(object sender, EventArgs e){
      LoadText(null);
}
```

Στην περίπτωση που το drag'n'drop μεταφέρει κάποιο απόσπασμα κειμένου, εισάγουμε το κείμενο στο σημείο που βρίσκεται ο κέρσορας. Ο κώδικας που το πετυχαίνει αυτό λειτουργεί όπως κι εκείνος που είχαμε δει για την αυτόματη εισαγωγή της ημερομηνίας και της ώρας.



Το DeltaPad υποστηρίζει πλέον και το drag'n'drop. Αποθέτοντας επάνω του ένα αρχείο κειμένου, το περιεχόμενό του φορτώνεται αυτόματα!

Απίθανη συμμαχία

Η C# είναι πανίσχυρη από μόνη της, αλλά οι κλάσεις που προσφέρει το .NET framework την απογειώνουν. Ένα από τα δυνατότερα χαρακτηριστικά του framework είναι η δυνατότητα "σύνδεσης" των εφαρμογών μας με προγράμματα ή και βιβλιοθήκες που είναι γραμμένες σε εγγενή κώδικα ή unmanaged code, όπως ονομάζεται στην ορολογία της πλατφόρμας .NET. (Να υπενθυμίσουμε εδώ ότι τα εκτελέσιμα που παράγονται από τον compiler του .NET δεν περιέχουν κώδικα σε γλώσσα μηχανής, αλλά ένα ενδιάμεσο bytecode γνωστό ως Common Intermediate Language. Αυτός ο κώδικας μετατρέπεται σε κώδικα μηχανής από το CLRuntime, κατά την εκτέλεση του προγράμματος. Καθώς η διαχείριση της μνήμης πραγματοποιείται από το CLR και όχι από τον προγραμματιστή, ο κώδικας CIL χαρακτηρίζεται ως "managed code".) Στη συνέχεια θα εξετάσουμε ένα απλό πρόγραμμα, το οποίο κάνει χρήση αυτής της δυνατότητας. Έτσι, θα δείτε με ποιον τρόπο μπορούμε να αξιοποιούμε άλλα προγράμματα ή βιβλιοθήκες των Windows.

Το πρόγραμμά μας διαθέτει δύο κουμπιά, μέσω των οποίων ανοίγει και κλείνει το πορτάκι του CD/DVD drive. Όσο κι αν ψάξετε στο.NET framework, δεν θα βρείτε καμία κλάση που να σχετίζεται με το θέμα και να προσφέρει αυτή τη λειτουργικότη-

τα. Μήπως πρέπει να σηκώσουμε τα χέρια ψηλά; Προφανώς, δεν θα ξεκινούσαμε όλη αυτή την εισαγωγή για να σας πούμε ότι τα παρατήσαμε. Το λειτουργικό σύστημα της Microsoft περιλαμβάνει από τα γεννοφάσκια του μια βιβλιοθήκη με το όνομα winmm.dll. Εκεί ενσωματώνονται οι συναρτήσεις του Windows Multimedia API, μέσω των οποίων μπορούμε να χειριστούμε την κάρτα ήχου, το joystick, κάθε συσκευή που συνδέεται στο midiport κ.ά. Ανάμεσα στις λειτουργίες που προσφέρει η βιβλιοθήκη περιλαμβάνεται και η αναπαραγωγή των audio CDs και, όπως ήταν αναμενόμενο, η εισαγωγή και η εξαγωγή του CD/DVD tray. Ειδικά γι' αυτές τις λειτουργίες αξιοποιούνται κάποιες εντολές MCI, αλλά δεν υπάρχει λόγος να μπούμε σε περισσότερες λεπτομέρειες.

Για την κατασκευή του προγράμματος δημιουργήσαμε ένα νέο project τύπου Windows Application και του δώσαμε το όνομα OpticalTray. Στη φόρμα τοποθετήσαμε δύο κουμπιά, με τα ονόματα buttonEject και buttonLoad. Νομίζουμε ότι η αποστολή του καθενός είναι αυτονόητη. Κοντά στα κουμπιά τοποθετήσαμε κι ένα Combo Box, στο οποίο δώσαμε το όνομα comboBoxLetters. Σε αυτό θα εμφανίζονται τα γράμματα του αγγλικού αλφαβήτου και ο χρήστης θα μπορεί να επιλέξει το επιθυμητό drive. Δείτε τον κώδικα που γράψαμε για αυτό το μικρό εργαλείο:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.Runtime.InteropServices;

namespace OpticalTray {
  public partial class MainForm : Form {
    [DllImport("winmm.dll")]
    static extern int mciSendString(String command, String buffer, int bufferSize, IntPtr hwndCallback);

   public MainForm() {
     InitializeComponent();

     comboBoxLetters.DropDownStyle = ComboBoxStyle.DropDownList;
     for (int i=0; i<26; i++ ) {
       comboBoxLetters.Items.Add((char)('A' + i) + ":");
     }
     comboBoxLetters.SelectedIndex = 3;
   }

   void ButtonEjectClick(object sender, EventArgs e) {
```

```
23      TrayDo(TrayOperation.Eject);
24    }
25
26    void ButtonLoadClick(object sender, EventArgs e) {
27      TrayDo(TrayOperation.Load);
28    }
29
30    private enum TrayOperation {
31      Eject,
32      Load
33    }
34
35    private void TrayDo(TrayOperation operation) {
36      string drive = (string) comboBoxLetters.Text;
37      string driveLetter = drive.Substring(0,1);
38      mciSendString("open " + driveLetter + ": type CDAudio alias drive" +
        driveLetter, null, 0, IntPtr.Zero);
39      if(operation == TrayOperation.Eject) {
40        mciSendString("set drive" + driveLetter + " door open", null, 0,
          IntPtr.Zero);
41      } else {
42        mciSendString("set drive"  + driveLetter + " door closed", null, 0,
          IntPtr.Zero);
43      }
44    }
45  }
46 }
```

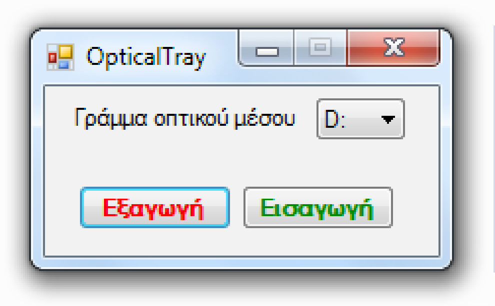
Μέχρι και την τέταρτη γραμμή είναι όλα γνωστά: Δηλώνονται τα απαραίτητα namespaces για τη δημιουργία της φόρμας και την αλληλεπίδραση με τα στοιχεία της. Στη γραμμή 5, όμως, εισάγουμε έναν χώρο ονομάτων που δεν είχαμε χρησιμοποιήσει στο παρελθόν. Πρόκειται για τον System.Runtime.InteropServices, που όπως συμπεραίνετε από το όνομά του παρέχει τις δυνατότητες διασύνδεσης με unmanaged code.

Στη γραμμή 9 δηλώνουμε τη χρήση της βιβλιοθήκης winmm.dll και, αμέσως μετά, προσδιορίζουμε το λεγόμενο entry point. Με απλά λόγια, καθορίζουμε το σημείο από το οποίο ξεκινά η εκτέλεση του κώδικα της βιβλιοθήκης. Όπως βλέπετε, εμείς επιλέγουμε τη συνάρτηση mciSendString. Παρατηρήστε τη χρήση του τροποποιητή extern, με τον οποίο δηλώνουμε ότι πρόκειται για μια συνάρτηση εκτός του .NET framework και εντός της βιβλιοθήκης που δηλώσαμε με την DllImport.

Όσον αφορά τα ορίσματα της mciSendString, αυτά δεν αποτελούν προϊόν τύχης αλλά μιας ενδελεχούς αναζήτησης. Αν σκοπεύετε να χρησιμοποιήσετε τέτοιες συναρτήσεις στα δικά σας προγράμματα, πρέπει προηγουμένως να ανατρέξετε στο Windows API και να βρείτε τις υπογραφές τους. Περισσότερα για τη δήλωση εξωτερικών βιβλιοθηκών και τα entry points μπορείτε να διαβάσετε στο https://msdn.microsoft.com/en-us/library/aa984739.aspx.

Ας προχωρήσουμε τώρα στη γραμμή 15, εντός της μεθόδου δημιουργίας του αντικειμένου MainForm. Εκεί αρχικοποιούμε τις τιμές του Combo Box. Αρχικά ορίζουμε την ιδιότητα DropDownStyle σε ComboBoxStyle.DropDownList, ώστε να έχουμε μια στατική λίστα. Ο χρήστης θα μπορεί να επιλέξει οποιαδήποτε από τις τιμές της λίστας, αλλά δεν θα μπορεί να τροποποιήσει καμία. Ακριβώς από κάτω (γραμμές 16 έως 18) ξεκινά ένας βρόχος που εισάγει τις επιλογές της λίστας (τα γράμματα της αλφαβήτου). Στη συνέχεια ορίζουμε ως προεπιλεγμένο στοιχείο το τρίτο, το οποίο αντιστοιχεί στο "D:" και είναι το γράμμα που αναφέρεται συνήθως στο CD/DVD drive.

Στη γραμμή 30 δηλώνουμε την απαρίθμηση TrayOperation, που περιλαμβάνει τα στοιχεία Load και Eject. Αυτή η απαρίθμηση χρησιμοποιείται ως τύπος της παραμέτρου που δέχεται η μέθοδος TrayDo(). Έτσι, οι κλήσεις προς τη συγκεκριμένη μέθοδο θα είναι ιδιαίτερα κομψές: TrayDo(Load) και TrayDo(Eject). Ας εξετάσουμε τώρα τον κώδικα της εν λόγω μεθόδου. Στις γραμμές 36 και 37 "αποσπάμε" το γράμμα από το επιλεγμένο στοιχείο του Combo Box και το αποθηκεύουμε στη μεταβλητή driveLetter. Στη συνέχεια (γραμμή 38) καλούμε για πρώτη φορά τη μέθοδο mciSendString.



Με τη βοήθεια της βιβλιοθήκης winmm.dll, το μικρό μας πρόγραμμα είναι στη θέση να ανοίξει και να κλείσει το συρταράκι του DVD drive. Ο ενθουσιασμός μας δεν έχει να κάνει με τον έλεγχο του DVD, αλλά με τη δυνατότητα αξιοποίησης των βιβλιοθηκών των Windows.

Ουσιαστικά, αυτή η κλήση είναι προπαρασκευαστική και δημιουργεί ένα alias που θα αξιοποιήσουμε αμέσως μετά, για το άνοιγμα ή το κλείσιμο του "συρταριού". Φυσικά, το περιεχόμενο των παραμέτρων συγκροτείται σύμφωνα με τις οδηγίες του Windows Multimedia API και δεν υπάρχει λόγος να επεκταθούμε σε αυτό το θέμα. Αμέσως μετά, η μέθοδος ελέγχει αν η τιμή της παραμέτρου operation είναι Eject ή Load. Σε κάθε περίπτωση πραγματοποιείται μια κατάλληλη κλήση της μεθόδου mciSendString. Αυτή η κλήση (η δεύτερη) πραγματοποιεί την αναμενόμενη ενέργεια και ανοίγει ή κλείνει το tray. Μπορείτε να κατεβάσετε τον κώδικα αυτού του μικρού προγράμματος από το http://bit.ly/dh048dnet10.

Δεν ξέρουμε αν το καταλάβατε, αλλά με τη βοήθεια των InteropServices μπορείτε να χρησιμοποιείτε το Windows API κανονικότατα. Πλέον, μπορείτε να πετύχετε στα Windows οτιδήποτε έκαναν και οι φίλοι που προγραμματίζουν σε C++. Εμείς από πλευράς μας, δεν έχουμε παρά να σας ευχηθούμε καλό ταξίδι!

Κατασκευάστε έναν υπολογιστή, μέρος 4:

GPU με απ' όλα

Το κύκλωμα γραφικών δεν ασχολείται αποκλειστικά με την παραγωγή του σήματος VGA. Η GPU έχει αποκλειστική πρόσβαση στη frame buffer κι αυτό την επιφορτίζει με αρκετά καθήκοντα. Οποιαδήποτε επέμβαση στην εικόνα, από τον καθαρισμό έως την εμφάνιση ενός χαρακτήρα ή μιας φωτογραφίας, αποτελεί ευθύνη του προγράμματός μας.

του Παναγιώτη Βαρελά

Στους πρώτους "προσωπικούς υπολογιστές" τα κυκλώματα γραφικών δεν ενσωμάτωναν κάποιο τσιπ για τη συμπίεση και την αποσυμπίεση βίντεο, ούτε Pixel Shader, ούτε Stream Processor, ούτε τα τεράστια ποσά μνήμης που συναντάμε στις σύγχρονες κάρτες γραφικών. Μη νομίζετε όμως ότι είχαν περιορισμένες αρμοδιότητες. Εκείνα τα κυκλώματα, όπως και το δικό μας, περιλάμβαναν ένα μικρό ποσό μνήμης για την αποθήκευση της εικόνας, επάνω στο οποίο δεν μπορούσε να γράψει κανένας άλλος. Ως εκ τούτου, οτιδήποτε εμφανιζόταν στην οθόνη είχε τοποθετηθεί στη frame buffer από το ίδιο το κύκλωμα γραφικών. Αυτό δεν είναι μικρό πράγμα! Οι GPU της εποχής είχαν την ευθύνη της σχεδίασης των χαρακτήρων και από αυτή την άποψη ενσωμάτωναν τη μοναδική γραμματοσειρά του συστήματος. Στην καλύτερη περίπτωση μπορεί να διέθεταν δύο ή τρεις γραμματοσειρές, στις οποίες άλλαζε μόνο το μέγεθος των γραμμάτων και η καθεμία προοριζόταν για μία από τις υποστηριζόμενες αναλύσεις. Αν αυτό δεν σας κάνει εντύπωση, σκεφτείτε ότι εκείνες οι GPU αναλάμβαναν και τη σχεδίαση του δρομέα (cursor), ενώ στη λίστα με τα προσόντα τους συναντούσε κανείς τη σχεδίαση απλών γεωμετρικών σχημάτων, όπως ευθύγραμμα τμήματα, τετράγωνα και κύκλους. Μάλιστα οι πιο προχωρημένες προσέφεραν και την (εκπληκτικής ισχύος) δυνατότητα του color flood. Δηλαδή μπορούσαν να γεμίσουν μια κλειστή περιοχή της εικόνας με ένα συγκεκριμένο χρώμα.

Πιστεύουμε ότι καταλαβαίνετε πού το πάμε: Παρόμοια προσόντα με τις παλιές GPU, διαθέτει και η δική μας. Τα καθήκοντά της δεν περιορίζονται στην παραγωγή των scanlines και κατ' επέκταση του σήματος VGA. Διαθέτει ένα πρωτόλειο API, μέσω του οποίου ο επεξεργαστής του υπολογιστή μπορεί να "παραγγείλει" τον καθαρισμό της οθόνης, την "εκτύπωση" ενός χαρακτήρα, την εμφάνιση ενός pixel, τη μετατόπιση του δρομέα και πάει λέγοντας.

Κώδικας επικοινωνίας

Όπως είδαμε στο άρθρο που αρχίζει από τη σελίδα 30, η κρισιμότερη εργασία του προγράμματος επιτελείται μέσα στη ρουτίνα scanline. Η συγκεκριμένη ρουτίνα εκτελείται σε τακτά χρονικά διαστήματα με τη βοήθεια ενός interrupt, το οποίο παράγεται από τον timer/counter1. Όλο τον υπόλοιπο χρόνο το πρόγραμμα βρίσκεται μέσα σ' έναν βρόχο και περιμένει εντολές από τον επεξεργαστή. Λαμβάνοντας υπόψη το μικρό πλήθος λειτουργιών που προσφέρει η GPU καθώς και τη χαμηλή ταχύτητα επικοινωνίας με τον επεξεργαστή, ορίσαμε ένα απλούστατο σύστημα επικοινωνίας. Κάθε εισερχόμενο byte με τιμή μικρότερη του 128 αντιμετωπίζεται ως κωδικός ASCII και το πρόγραμμα φροντίζει να εμφανίσει στην οθόνη τον αντίστοιχο χαρακτήρα. Βέβαια, στη συγκεκριμένη περιοχή τιμών συγκαταλέγονται και οι χαρακτήρες ελέγχου (LINE FEED, CARRIAGE RETURN, BACKSPACE κ.ά.), οι οποίοι δεν είναι εκτυπώσιμοι. Το πρόγραμμα προβλέπει τη λήψη τέτοιων χαρακτήρων και μπορεί να μετατοπίζει το δρομέα στην κατάλληλη θέση, να διαγράφει χαρακτήρες κ.λπ. Όσα bytes στέλνει ο επεξεργαστής κι έχουν τιμή μεγαλύτερη του 127 αντιμετωπίζονται ως εντολές. Κάποιες από αυτές, όπως ο καθαρισμός της οθόνης και η "επανεκκίνηση" του κυκλώματος γραφικών, δεν δέχονται παραμέτρους. Υπάρχουν όμως κι εντολές, όπως εκείνη για την αλλαγή χρώματος των χαρακτήρων, που συνοδεύονται πάντα από παραμέτρους. Όταν λαμβάνονται εντολές της δεύτερης κατηγορίας το πρόγραμμα αντιμετωπίζει τα επόμενα bytes ως παραμέτρους και δεν πραγματοποιεί το διαχωρισμό σε χαρακτήρες και εντολές. Μετά τη λήψη των παραμέτρων και αφού εκτελεσθεί η εντολή, η ροή του προγράμματος επιστρέφει στον κεντρικό βρόχο και περιμένει νέες εντολές και νέους χαρακτήρες.

■ FEATURES

- Capable of controlling both LCD and CRT displays.
- Includes 6845 restricted mode and CRT peripheral circuits for IMB-PC.
- Both SRAM and DRAM are usable as VRAM.
- Includes MOUSE and LIGHT PEN interface.
- Cursor position can be specified by any 16 × 16 dot patterns in the bit unit (AND and EXOR screens).
- Includes color palette (16/512 colors).
- LCD intensity controllable (16 or 8 gradation steps)
- Screen modes are available in combinations of the following.
  - Horizontal dot number: 640, 320, 512, 256
  - Vertical dot number: 192, 200, 204, 64 (64 only with LCD)
  - Raster adjustment: 0, 2, 4 or 6 specifiable
- Capable of displaying 16 colors in 640 × 204 by using external circuits.
- CRT monitor selectable from among IBM Color, Monochrome, NTSC system and PAL system.
- Can be interfaced with 3 types of LCD driver.
- Usable with 16 bit bus CPU.

NIPPON GAKKI CO., LTD.

V6355D CATALOG
CATALOG No. : LSI-216355D
1987.06

1

Τα προσόντα μιας προϊστορικής GPU: Προσφέρει διάφορες αναλύσεις, αναλαμβάνει τη διαχείριση του δρομέα και ενσωματώνει μέχρι και interface για ποντίκι και light pen. Τα τσιπάκια γραφικών εκείνης της εποχής ήταν πολυτάλαντα, αν και δεν ήταν ιδιαίτερα αποδοτικά στο χειρισμό διανυσματικών μεγεθών ;)

Πολλαπλές διακλαδώσεις

Κάθε φορά που ο επεξεργαστής στέλνει "απλούς" χαρακτήρες, εκτελείται η ρουτίνα εκτύπωσης. Οι εντολές, ωστόσο, δεν έχουν υλοποιηθεί σε μία ενιαία ρουτίνα. Η κοινή λογική επέβαλε το διαχωρισμό τους σε επιμέρους τμήματα κώδικα, ώστε το πρόγραμμα να είναι ευανάγνωστο και διαχειρίσιμο. Έτσι, κάθε φορά που καταφτάνει

μια εντολή, το πρόγραμμα πρέπει να καλέσει την αντίστοιχη ρουτίνα. Τώρα μπορεί να αναρωτιέστε γιατί συζητάμε κάτι τόσο τετριμμένο. Θα μπορούσαμε να χρησιμοποιήσουμε μια δομή σαν τη switch και να ξεμπερδέψουμε αμέσως. Στην Assembly όμως δεν υπάρχει τίποτα παραπλήσιο. Μια εναλλακτική λύση θα ήταν να συνδυάσουμε μερικούς απλούς ελέγχους:

```
if (received_byte < 128)
    call print_character
else if (received_byte == 200)
    call command1
else if (received_byte == 201)
    call command2
else if (received_byte == 202)
    call command3
...
```

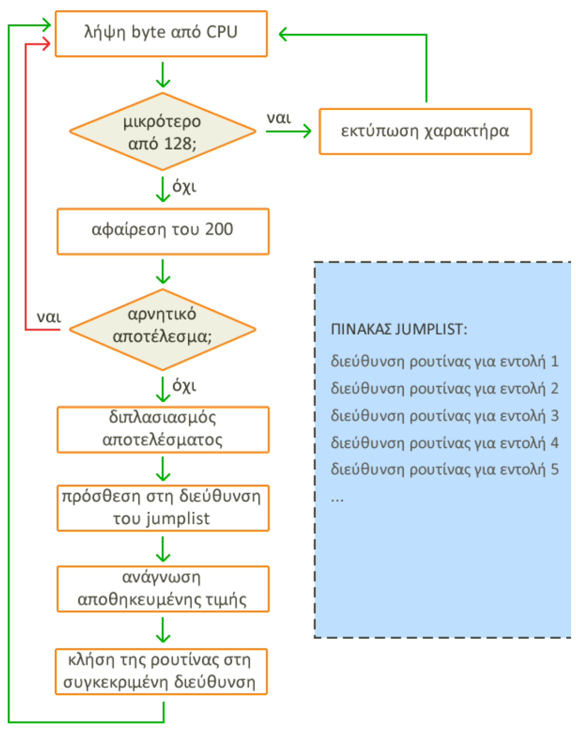
Ο παραπάνω ψευδοκώδικας είναι ευανάγνωστος, αλλά όχι κι αποδοτικός. Όσο αυξάνεται το πλήθος των εντολών και συνεπώς το πλήθος των ελέγχων, εισάγονται μεγαλύτερες καθυστερήσεις. Φανταστείτε τώρα να γράφαμε κάτι τέτοιο σε Assembly. Επειδή η εντολή "else if" δεν υπάρχει, στην περίπτωση που επαληθευόταν ένας από τους ελέγχους θα έπρεπε να φροντίσουμε εμείς για την παράκαμψη των υπολοίπων. Κάτι τέτοιο θα μπορούσε να γίνει τοποθετώντας "κάτω" από κάθε έλεγχο ένα μικρό μπλοκ κώδικα, με την κλήση της αντίστοιχης ρουτίνας και μία εντολή για την παράκαμψη των περιττών ελέγχων. Αυτό δεν ακούγεται και πολύ κακό, αλλά υπάρχει ένα μικρό πρόβλημα: Στην Assembly δεν υφίσταται η έννοια του μπλοκ κώδικα και το ζήτημα περιπλέκεται ακόμα περισσότερο. Όπως καταλαβαίνετε, υλοποιώντας την παραπάνω λύση σε Assembly το πρόγραμμα θα καταντούσε δυσανάγνωστο. Για αυτούς τους λόγους κι επειδή μιλάμε για ένα κομβικό τμήμα του προγράμματος, καταφύγαμε στη λύση των jumplists.

Ο δείκτης του δείκτη

Ας ξαναδούμε εν συντομία το πρόβλημα. Κάθε φορά που η GPU λαμβάνει μια εντολή, το πρόγραμμα πρέπει να καλέσει την αντίστοιχη ρουτίνα. Αν βρίσκαμε κάποιον τρόπο για να συσχετίσουμε τις ίδιες τις εντολές (τις τιμές των bytes) με τις διευθύνσεις των αντίστοιχων ρουτινών, το πρόβλημα θα εξαφανιζόταν. Το πρόγραμμα θα υπολόγιζε τη διεύθυνση της κατάλληλης ρουτίνας και θα την καλούσε, χωρίς να πραγματοποιηθεί μια ορδή ελέγχων. Τα jumplists εξυπηρετούν στην κατασκευή ακριβώς ενός τέτοιου μηχανισμού. Πρόκειται για απλούς πίνακες, κάθε στοιχείο των οποίων αποτελεί τη διεύθυνση εκκίνησης μιας συνάρτησης. Το jumplist για το δικό μας πρόγραμμα ορίστηκε ως εξής:

```
jumplist:
.dw reset   ; index 00
.dw clear   ; index 01
.dw pset    ; index 02
.dw line    ; index 03
.dw box     ; index 04
.dw locate  ; index 05
...
```

Το ".dw" αποτελεί οδηγία προς τον assembler και όχι εντολή του μικροελεγκτή. Το αποτέλεσμά της είναι η αποθήκευση ενός word (δύο bytes) στη μνήμη προγράμματος. Οι λέξεις reset, clear, pset κ.λπ. αποτελούν τα ονόματα των ρουτινών που έχουμε κατασκευάσει για την εκτέλεση κάθε εντολής. Αυτά τα ονόματα αντικαθίστανται αυτομάτως με τις διευθύνσεις μνήμης των ρουτινών. Επειδή ο μικροελεγκτής που χρησιμοποιούμε διαθέτει 64KB για την αποθήκευση του κώδικα, οι διευθύνσεις έχουν μήκος ακριβώς 16 bits (δύο bytes). Τελικά, οι παραπάνω γραμμές δημιουργούν έναν πίνακα με τις διευθύνσεις εκκίνησης των ρουτινών. Τώρα μένει να δούμε πώς τον αξιοποιήσαμε. Ούτε κι αυτό είναι δύσκολο, αλλά θέλει λίγη προσοχή παραπάνω.



Ο κυρίως βρόχος του προγράμματος. Η GPU ξοδεύει ένα μεγάλος μέρος του χρόνου της εδώ, περιμένοντας από τον κεντρικό επεξεργαστή εντολές και χαρακτήρες προς εκτύπωση. Για την εκτέλεση των εντολών αξιοποιείται ένα jumplist.

Κατ' αρχάς, επειδή η GPU υποστηρίζει μόλις δώδεκα εντολές αποφασίσαμε να τις αντιστοιχίσουμε στις τιμές από 200 έως και 211. Έτσι, όταν το πρόγραμμα λαμβάνει κάποιο byte που δεν αντιστοιχεί σε χαρακτήρα, αφαιρεί από την τιμή του το 200. Αν το αποτέλεσμα είναι αρνητικό, θεωρούμε ότι συνέβη κάποιο σφάλμα και το συγκεκριμένο byte αγνοείται (το πρόγραμμα επιστρέφει στην αρχή του κυρίως βρόχου):

```
subi data, 200          ; subtract 200
brlo main_loop
```

Αν το αποτέλεσμα της αφαίρεσης είναι θετικό, το θεωρούμε αυτόματα ως τον αύξοντα αριθμό της επιθυμητής εντολής. Το επόμενο βήμα είναι να διαβάσουμε το αντίστοιχο στοιχείο από τον πίνακα jumplist. Για το σκοπό αυτό αποθηκεύουμε τη διεύθυνση εκκίνησης του πίνακα στον καταχωρητή Z. Προσέξτε τώρα το εξής: Κάθε στοιχείο του εν λόγω πίνακα έχει μήκος δύο bytes. Συνεπώς, για να διαβάσουμε το πέμπτο στοιχείο πρέπει να προσθέσουμε στον καταχωρητή Z το 10, για να διαβάσουμε το τρίτο στοιχείο πρέπει να προσθέσουμε το 6 κ.ο.κ. Με άλλα λόγια, πρέπει να διπλασιάζουμε τον αύξοντα αριθμό του επιθυμητού στοιχείου και να προσθέτουμε το αποτέλεσμα στον Z. Αυτό ακριβώς επιτυγχάνουν οι ακόλουθες γραμμές:

```
ldi ZH, high(jumplist*2) ; get jumplist starting address
ldi ZL, low(jumplist*2)  ;
lsl data                 ; double the offset (table contains words!)
add ZL, data             ; add offset to low byte jumplist pointer
adc ZH, zero             ;
```

Ο διπλασιασμός του αύξοντα αριθμού της εντολής πραγματοποιείται στην τρίτη γραμμή. Η εντολή lsl (logic shift left) ολισθαίνει τα bit ενός καταχωρητή προς τα αριστερά και στη θέση του πρώτου εισάγει ένα μηδενικό. Με αυτόν τον τρόπο η τιμή του καταχωρητή διπλασιάζεται με τον πιο γρήγορο τρόπο. Αν χρησιμοποιούσαμε την εντολή πολλαπλασιασμού, θα έπρεπε να αποθηκεύσουμε τον πολλαπλασιαστή (2) σε κάποιον πρόσθετο καταχωρητή κι αυτό θα συνιστούσε "πολλή φασαρία για το τίποτα". Με όσα έχουμε δει ως τώρα το πρόγραμμα υπολογίζει τη διεύθυνση ενός στοιχείου του πίνακα jumplist, που αντιστοιχεί στην προς εκτέλεση εντολή. Αυτό το στοιχείο, με τη σειρά του, αποτελεί τη διεύθυνση της αντίστοιχης συνάρτησης και απαρτίζεται από δύο bytes. Ο κώδικας συνεχίζει με την ανάγνωση αυτών των bytes και την αποθήκευσή τους στον καταχωρητή Z:

```
lpm tmp1, Z+             ; store low-byte of address-to-call in tmp1
lpm ZH, Z                ; get high-byte of address-to-call in ZH
mov ZL, tmp1             ; move tmp1 to ZL
```

Αυτό ήταν όλο! Ο Z περιλαμβάνει πλέον τη διεύθυνση της ρουτίνας που πρέπει να εκτελεστεί και αρκεί να πραγματοποιήσουμε την κλήση.

```
    icall                       ; call routine at address Z
```

Ο μηχανισμός που στήσαμε γύρω από τον πίνακα jumplist επιτρέπει την εύκολη προσθαφαίρεση εντολών, χωρίς να αλλάζει η έκταση ή ο χρόνος εκτέλεσης του κώδικα. Αρκεί οι διευθύνσεις των ρουτινών που έχουμε τοποθετήσει στον πίνακα να βρίσκονται σε αντιστοιχία με τις αριθμητικές τιμές των εντολών. Για παράδειγμα, αν η δεύτερη εντολή πραγματοποιεί τον καθαρισμό την οθόνης, το δεύτερο στοιχείο του πίνακα jumplist πρέπει να είναι η διεύθυνση εκκίνησης της ρουτίνας που καθαρίζει την οθόνη. Δεν ξέρουμε τι πιστεύετε εσείς, αλλά το σημαντικότερο προσόν αυτού του μηχανισμού θεωρούμε ότι σχετίζεται με την κομψότητα ;)

Εσωτερική ισορροπία

Η ρουτίνα scanline "σπάει το κοντέρ" κατά τη διάρκεια του ορατού τμήματος της γραμμής. Σε αυτή τη φάση πετυχαίνει το μέγιστο pixel clock που επιτρέπουν τα χαρακτηριστικά του μικροελεγκτή και δεν της μένει χρόνος για τίποτε άλλο. Αυτό δεν σημαίνει ότι τα πράγματα είναι χαλαρά κατά τη διάρκεια του παλμού HSYNC ή των διαστημάτων back porch και front porch. Αν εξετάσετε τον κώδικα θα διαπιστώσετε ότι εντός αυτών των διαστημάτων υπάρχουν αρκετά nop (εντολές που δεν κάνουν τίποτα κι απλά καθυστερούν την εκτέλεση κατά έναν κύκλο). Αναρωτιέστε γιατί προστέθηκαν και γιατί δεν βάλαμε στη θέση τους κάτι άλλο, περισσότερο χρήσιμο; Σκεφτείτε ότι ο κώδικας περιλαμβάνει αρκετές διακλαδώσεις. Αυτό σημαίνει ότι η εκτέλεσή του δεν ακολουθεί πάντοτε την ίδια πορεία: Κάποιες φορές εκτελείται το ένα τμήμα μιας διακλάδωσης και κάποιες φορές το άλλο. Όπως έχουμε πει, όμως, η ρουτίνα scanline πρέπει να εκτελείται με απόλυτα σταθερή συχνότητα κι αυτός ο "νόμος" ισχύει και για το εσωτερικό της — για όλα τα επιμέρους τμήματα της γραμμής. Για να πετύχουμε την "εσωτερική" σταθερότητα πρέπει να εξασφαλίσουμε ότι όλες οι πιθανές πορείες εκτέλεσης έχουν την ίδια διάρκεια. (Σ.τ.Ε. Εγώ πάλι νόμιζα ότι η εσωτερική σταθερότητα διασφαλίζεται με την ανάπτυξη...) Με άλλα λόγια, πρέπει να εξετάσουμε όλες τις διακλαδώσεις και για καθεμία να μετρήσουμε το χρόνο εκτέλεσης κάθε σκέλους. Αν κάποιο από τα δύο είναι γρηγορότερο πρέπει να το καθυστερήσουμε τόσο, ώστε να διαρκεί όσο το άλλο. Εδώ ακριβώς βοηθάνε τα nop. Κι επειδή οι συγκεκριμένες εντολές δεν εκτελούνται πάντα, είναι πρακτικά αδύνατο να αντικατασταθούν από άλλες, που θα έκαναν κάτι χρήσιμο.

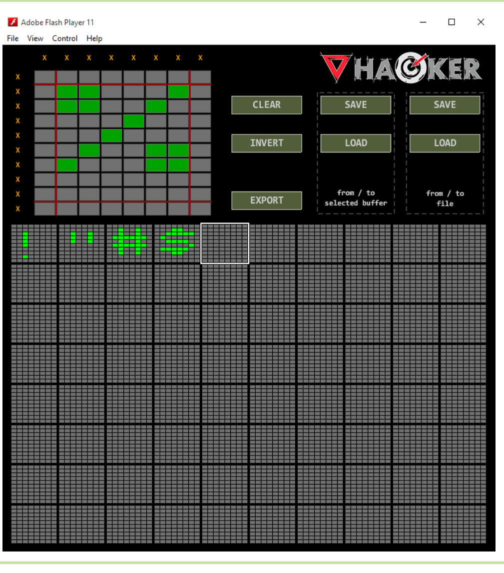
Χειρισμός χαρακτήρων

Έχουμε αναφερθεί αρκετές φορές στη ρουτίνα εκτύπωσης, αλλά μέχρι στιγμής δεν έχουμε πει τίποτα για τον κώδικά της. Μιλάμε για τη ρουτίνα putchar, που καλείται από τον κύριο βρόχο του προγράμματος κάθε φορά που καταφτάνει κάποιος χαρακτήρας. Μπορείτε να δείτε τον κώδικά της στο αρχείο 1.prep.asm. Όπως θα

διαπιστώσετε, η συγκεκριμένη ρουτίνα πραγματοποιεί έναν βασικό έλεγχο για το αν ο εισερχόμενος χαρακτήρας είναι εκτυπώσιμος ή όχι. Στην πρώτη περίπτωση καλείται η ρουτίνα print_char, ενώ στη δεύτερη καλείται η ρουτίνα print_ctrl. Η πρώτη σχεδιάζει τον εκάστοτε χαρακτήρα στη frame buffer, ενώ η δεύτερη πραγματοποιεί τη λειτουργία που περιγράφει ο εκάστοτε χαρακτήρας ελέγχου. Αυτές οι δύο ρουτίνες επιδρούν στο περιεχόμενο της frame buffer και τις έχουμε τοποθετήσει στο αρχείο 2.buffer.asm.

Συνήθως, η print_ctlr περιορίζεται σε μια μετακίνηση του δρομέα και σπανίως τροποποιεί την εικόνα. Ακόμα κι όταν συμβαίνει κάτι τέτοιο, π.χ., για τη διαγραφή ενός χαρακτήρα, επιστρατεύεται η print_char για την εκτύπωση ενός κενού στην κατάλληλη θέση. Όπως αντιλαμβάνεστε, η ρουτίνα print_ctrl δεν παρουσιάζει ιδιαίτερο ενδιαφέρον. Επιπρόσθετα, καθώς οι κωδικοί ASCII των χαρακτήρων ελέγχου δεν έχουν διαδοχικές τιμές, η χρήση ενός jumplist θα περιέπλεκε τα πράγματα περισσότερο. Για αυτό το λόγο καταλήξαμε να γράψουμε ένα κατεβατό από διαδοχικούς ελέγχους. Αν εξετάσετε τον κώδικα, πιστεύουμε ότι θα συμφωνήσετε αμέσως με την εκτίμηση που εκφράσαμε νωρίτερα: οι "αλυσιδωτοί" έλεγχοι στην Assembly προκαλούν σύγχυση.

Τελικά, το περισσότερο ενδιαφέρον συγκεντρώνεται στη ρουτίνα print_char. Για τη GPU, κάθε χαρακτήρας αποτελεί μια μικροσκοπική εικόνα. Επομένως, κάθε φορά που λαμβάνεται ένας εκτυπώσιμος χαρακτήρας η ρουτίνα print_char αντιγράφει την αντίστοιχη εικόνα στη frame buffer. Οι εν λόγω εικόνες έχουν διαστάσεις 8x10 pixels (πλάτος x ύψος) και είναι μονόχρωμες. Αυτό σημαίνει ότι κάθε pixel αναπαρίσταται ακριβώς από ένα bit και, κατ' επέκταση, κάθε γραμμή αναπαρίσταται από ένα byte. Έτσι, η όψη κάθε χαρακτήρα περιγράφεται από 10 bytes. Όλες αυτές οι εικόνες βρίσκονται αποθηκευμένες στον πίνακα fontdata ο οποίος, όπως αντιλαμβάνεστε, αποτελεί τη γραμματοσειρά του υπολογιστή μας. Μπορείτε να δείτε το περιεχόμενο του πίνακα στο αρχείο 3.font.asm.



Η σχεδίαση της γραμματοσειράς πραγματοποιήθηκε χαρακτήρα προς χαρακτήρα, σε ένα "ανακυκλωμένο" πρόγραμμα.

Αν μελετήσετε τον κώδικα που εμφανίζει τους χαρακτήρες στην οθόνη, θα διαπιστώσετε ότι περιλαμβάνει δυο εμφωλευμένους βρόχους. Ο εξωτερικός (ξεκινά με την ετικέτα get_font_byte) διατρέχει τα δέκα bytes που απαρτίζουν την όψη του προς εκτύπωση χαρακτήρα. Ο εσωτερικός βρόχος (ξεκινά με την ετικέτα parse_bits) εξετάζει ένα προς ένα τα bits του εκάστοτε byte. Όταν κάποιο bit έχει την τιμή 1, τοποθετείται στη frame buffer ένα pixel με το χρώμα που περιγράφει ο καταχωρητής color. Στην αντίθετη περίπτωση τοποθετείται ένα pixel με το χρώμα που περιγράφει ο καταχωρητής paper. Με αυτόν τον τρόπο, η όψη κάθε χαρακτήρα μεταφέρεται στην εικόνα γραμμή προς γραμμή. Οι καταχωρητές color και paper χρησιμοποιούνται σαν δύο μεταβλητές global και το περιεχόμενό τους αποτελεί το χρώμα των γραμμάτων και το χρώμα του υποβάθρου αντίστοιχα. Παρεμπιπτόντως, σημειώστε ότι η GPU περιλαμβάνει δύο εντολές που επιτρέπουν στον επεξεργαστή να τροποποιήσει αυτά τα χρώματα.

Πριν προχωρήσουμε πρέπει να απαντήσουμε σ' ένα ερώτημα που ενδέχεται να σας προβληματίζει: Πώς σχεδιάσαμε την όψη των χαρακτήρων; Γι' αυτή τη δουλειά αξιοποιήσαμε ένα πρόγραμμα που είχαμε κατασκευάσει στο μακρινό παρελθόν. Αναφερόμαστε σ' ένα υποτυπώδες σχεδιαστικό εργαλείο, με το οποίο δημιουργούσαμε "γραφικά" για ένα μονόχρωμο dot matrix 8x8. Το πρόγραμμα ήταν γραμμένο σε flash και παρόλο που τώρα δεν συμπαθούμε καθόλου τη συγκεκριμένη πλατφόρμα, θα ήταν κρίμα να πετάξουμε τόση δουλειά. Στο κάτω κάτω, η σχεδίαση των χαρακτήρων θα γινόταν μόνο μία φορά. Έτσι, με ελάχιστες επεμβάσεις μετατρέψαμε το παλιό πρόγραμμα σε εργαλείο σχεδίασης χαρακτήρων. Αν θέλετε να παίξετε κι εσείς μαζί του μπορείτε να το κατεβάσετε από το http://bit.ly/dh048fontedit.

Ξεζούμισμα

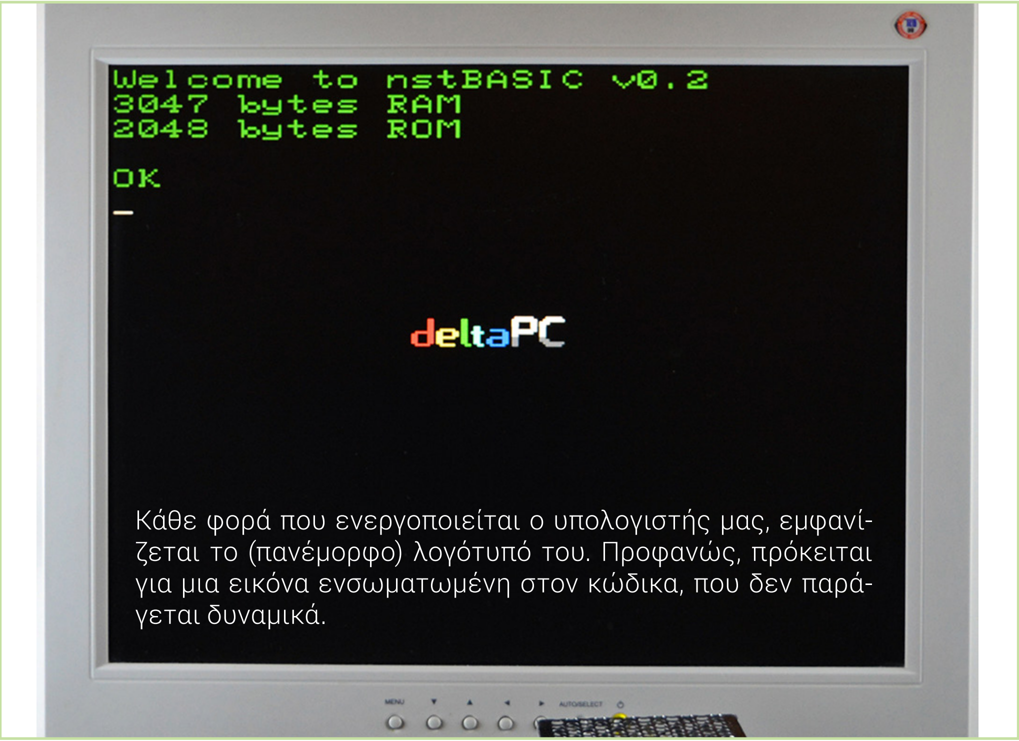
Η ρουτίνα scanline εκτελείται με μεγάλη συχνότητα και διακόπτει όλες τις υπόλοιπες εργασίες, συνεχώς. Γι' αυτό το λόγο, μία από τις πρώτες ενέργειες μέσα στη ρουτίνα είναι η αποθήκευση της κατάστασης των ακροδεκτών της θύρας PORTB. Όπως φαίνεται και στο κύκλωμα, οι συγκεκριμένοι ακροδέκτες χρησιμοποιούνται για την ανταλλαγή κάποιων σημάτων με τον κεντρικό επεξεργαστή, ρυθμίζουν τη λειτουργία της frame buffer, ενώ ελέγχουν και τη ροή των δεδομένων στο data bus. Εν ολίγοις, η κατάσταση αυτών των ακροδεκτών αποτυπώνει την γενική κατάσταση της GPU και συνδέεται στενά με όλες τις λειτουργίες του κυκλώματος. Αμέσως μετά αποθηκεύεται και η τιμή του δείκτη Z, που ενδέχεται να ήταν σε χρήση. Λίγο αργότερα, αν βρισκόμαστε σε ορατή γραμμή, που σημαίνει ότι θα χρειαστεί να διαβαστούν δεδομένα από τη frame buffer, αποθηκεύεται και η κατάσταση των ακροδεκτών των υπολοίπων θυρών. Φυσικά, λίγο πριν τερματιστεί η ρουτίνα scanline πραγματοποιείται η αντίστροφη διαδικασία: Οι τιμές που είχαν αποθηκευτεί νωρίτερα τοποθετούνται και πάλι στις αντίστοιχες θέσεις. Έτσι, με τον τερματισμό της ρουτίνας ο μικροελεγκτής μπορεί να συνεχίσει την προηγούμενη εργασία του κανονικά, σαν μην έγινε η διακοπή ποτέ. Σε ένα οποιοδήποτε άλλο πρόγραμμα, η προσωρινή αποθήκευση των τιμών θα γινόταν στη στοίβα. Στο πρόγραμμα που εξετάζουμε, όμως, ακολουθήσαμε μια διαφορετική οδό. Οι 8μπιτοι μικροελεγκτές AVR ενσωματώνουν στον πυρήνα τους 32 καταχω-

ρητές, για την προσωρινή αποθήκευση δεδομένων. Μιλάμε για τον πιο πολύτιμο πόρο των μικροελεγκτών, αφού οι συγκεκριμένοι καταχωρητές μπορούν να λάβουν μέρος άμεσα, σε κάθε πιθανή πράξη. Ωστόσο, εμείς αποφασίσαμε εξαρχής ότι θα αξιοποιούσαμε μόνο τους 22. Με αυτή τη γενναία σχεδιαστική επιλογή θέλαμε να εξασφαλίσουμε ότι μέσα στη ρουτίνα scanline θα υπάρχουν πάντα δέκα "ελεύθεροι" καταχωρητές. Σε αυτούς θα αποθηκεύαμε προσωρινά τις διάφορες τιμές και θα αποφεύγαμε τη στοίβα εντελώς. Αναρωτιέστε γιατί μπήκαμε σε αυτό το μπελά; Οι εντολές push και pop (προσθήκη και απομάκρυνση ενός byte από τη στοίβα) διαρκούν από δύο κύκλους η καθεμία. Αντίθετα, οι εντολές mov και movw (μετακίνηση ενός byte κι ενός word μεταξύ καταχωρητών) εκτελούνται σε έναν κύκλο η καθεμία. Καθώς η ρουτίνα scanline αποθηκεύει και επαναφέρει δέκα bytes, με τη μέθοδο που ακολουθήσαμε καταφέραμε να κερδίσουμε 20 κύκλους! Κι όπως αντιλαμβάνεστε, δεν μπορούσαμε να χαραμίσουμε κανέναν.

Ταχύτατη κύλιση

Όταν ολοκληρώσαμε τον κώδικα για την εμφάνιση των χαρακτήρων, ήρθαμε αντιμέτωποι με ένα νέο ζήτημα. Τι θα γινόταν όταν γέμιζε η οθόνη με χαρακτήρες ή, τέλος πάντων, όταν θα τυπωνόταν ένας χαρακτήρας στην τελευταία στήλη της τελευταίας γραμμής; Το λογικό θα ήταν να "ρολάρει" η εικόνα προς τα πάνω: να μετατοπιστούν τα περιεχόμενά της κατά μία γραμμή κειμένου προς τα πάνω και στο κάτω άκρο να εμφανιστεί μια κενή γραμμή. Για το σκοπό αυτό γράψαμε τη ρουτίνα roll, που στην αρχική της μορφή ήταν αρκετά μεγάλη και βραδυφλεγής! Βλέπετε, η πρώτη προσέγγιση στο ζήτημα ήταν αφελής. Ο κώδικας αντέγραφε κάθε pixel της frame buffer λίγες γραμμές πιο πάνω. Οι αντιγραφές ξεκινούσαν από τη 11η γραμμή pixel, η οποία κατέληγε στη θέση της 1ης και τερματίζονταν στην 240η, που κατέληγε στη θέση της 230ης. Ακολούθως, οι τελευταίες δέκα γραμμές pixel, που αντιστοιχούν στην τελευταία γραμμή κειμένου (την 24η), διαγράφονταν. Μ' αυτά και με τ' άλλα, το πρόγραμμα τροποποιούσε *όλα* τα pixels της εικόνας. Όπως αντιλαμβάνεστε, οι σχετικές επεμβάσεις στη frame buffer καθυστερούσαν πάρα πολύ και προέκυπτε ένα φαινόμενο που είναι γνωστό σαν screen tearing. Ένα τμήμα της εικόνας εμφανιζόταν να έχει κυλίσει προς τα επάνω, ενώ η υπόλοιπη εμφανιζόταν στην αρχική της θέση. Μάλιστα, το πρόβλημα ήταν τόσο έντονο, που το "σπάσιμο" της εικόνας φαινόταν να ταξιδεύει σαν ένας κυματισμός!

Η ανάγνωση και η εγγραφή ολόκληρης της frame buffer απαιτεί εκ των πραγμάτων πολύ χρόνο και καμία βελτιστοποίηση δεν μπορούσε να σώσει την κατάσταση. Έπρεπε να βρούμε μια ριζικά διαφορετική λύση, η οποία θα απαιτούσε λιγότερες επεμβάσεις στην εικόνα. Η αλήθεια είναι ότι για κάμποσους μήνες παριστάναμε ότι η αργή κύλιση της εικόνας δεν μας ενοχλεί. Ωστόσο ήταν αδύνατο να ξεχάσουμε στ' αλήθεια το πρόβλημα και σε μια αναλαμπή ευρηματικότητας σκεφτήκαμε το τέλειο τέχνασμα: Φανταστείτε την οθόνη σαν ένα παράθυρο, μέσα από το οποίο βλέπουμε μια περιοχή της μνήμης γραφικών. Η αρχική λύση για την κύλιση της εικόνας, προέβλεπε τη μετατόπιση όλων των δεδομένων της μνήμης. Η ιδανική λύση, όμως, θα ήταν να μετατοπίζαμε το παράθυρο!



Κάθε φορά που ενεργοποιείται ο υπολογιστής μας, εμφανίζεται το (πανέμορφο) λογότυπό του. Προφανώς, πρόκειται για μια εικόνα ενσωματωμένη στον κώδικα, που δεν παράγεται δυναμικά.

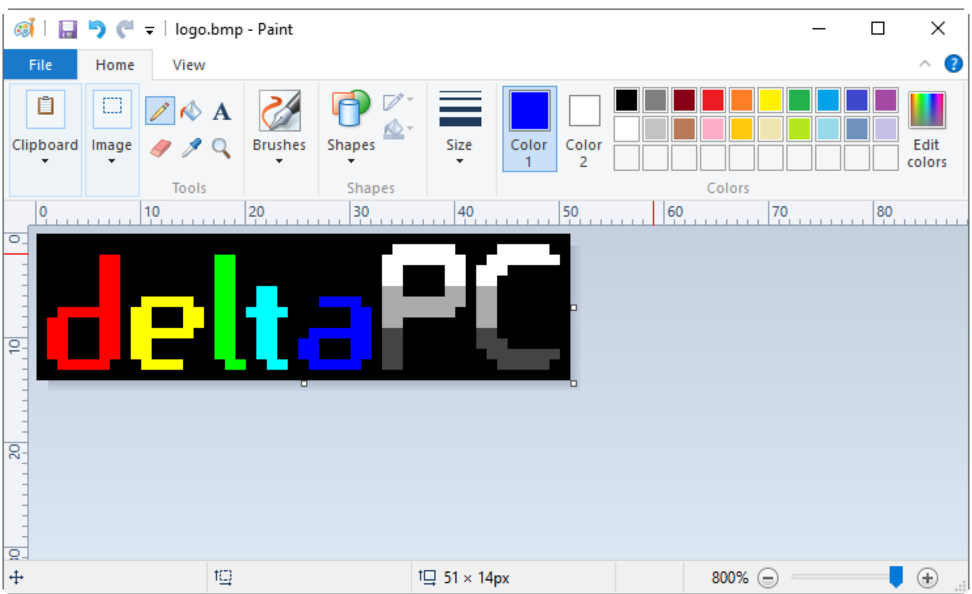
Όπως είδαμε στο προηγούμενο άρθρο, η κατακόρυφη ανάλυση της εικόνας ανέρχεται στα 240 pixels. Ακριβώς γι' αυτό, η ρουτίνα scanline στέλνει στην οθόνη το περιεχόμενο 240 γραμμών. Η αρχική εκδοχή της ρουτίνας scanline ξεκινούσε πάντα από την *πρώτη* γραμμή της frame buffer. Για να πετύχουμε το κολπάκι με το "παράθυρο" τροποποιήσαμε τη ρουτίνα κατάλληλα, ώστε να μην ξεκινά υποχρεωτικά από την πρώτη γραμμή. Πλέον, λίγο πριν ξεκινήσει το ορατό τμήμα ενός scanline, ο κώδικας προσθέτει στον αύξοντα αριθμό της γραμμής –στο high-byte του address bus– μια τιμή που λειτουργεί ως offset. Αναφερόμαστε στην τιμή της μεταβλητής sl_offset. Με αυτή τη μικρή τροποποίηση του προγράμματος, κάθε φορά που θέλουμε να κυλίσει η εικόνα κατά μία γραμμή κειμένου προς τα πάνω, αρκεί να αυξήσουμε την τιμή της sl_offset κατά δέκα. Το επόμενο frame θα σχηματιστεί από ένα διαφορετικό τμήμα της μνήμης οθόνης και η κύλιση της εικόνας θα πραγματοποιηθεί ακαριαία – για την ακρίβεια στο ένα εξηκοστό του δευτερολέπτου! Βέβαια, αμέσως μετά τη μετατόπιση της εικόνας πρέπει να καθαρίσουμε τις δέκα χαμηλότερες γραμμές pixels, που απαρτίζουν την τελευταία γραμμή κειμένου. Ωστόσο αυτή η επέμβαση στη μνήμη οθόνης ολοκληρώνεται σε ελάχιστο χρόνο και η κύλιση παραμένει ταχύτατη.

Κάπου εδώ μπορεί να προκύψει μια νέα απορία: Τι θα συμβεί όταν η τιμή της sl_offset ανέλθει, λόγου χάρη, στο 60, και η ρουτίνα scanline προσπαθήσει να διαβάσει την 200η γραμμή της εικόνας; Τίποτα το ανησυχητικό! Σε έναν οκτάμπιτο μικροελεγκτή, η πρόσθεση του 60 με το 200 έχει σαν αποτέλεσμα την ενεργοποίηση της σημαίας υπερχείλισης, ενώ παράγεται και ο αριθμός 4. Το πρόγραμμά μας αδιαφο-

ρεί πλήρως για την υπερχείλιση, κρατάει το 4 και αρχίζει να διαβάζει την τέταρτη γραμμή pixels της μνήμης οθόνης. Με άλλα λόγια, η ροή του προγράμματος δεν θα διαταραχθεί καθόλου. Βέβαια, η χρήση του offset έχει σαν αποτέλεσμα τη δέσμευση περισσότερου χώρου στη μνήμη οθόνης. Η frame buffer μοιάζει να κινείται μέσα στη μνήμη οθόνης κι απαιτεί μια έκταση που χωράει 256 γραμμές pixels και όχι 240. Λαμβάνοντας υπόψη ότι ακόμα κι έτσι περισσεύει άφθονος χώρος στο τσιπάκι μνήμης, αυτό το τίμημα κρίθηκε αμελητέο.

Μετατροπές χρωμάτων

Κάθε φορά που ενεργοποιείται ο υπολογιστής μας η οθόνη προβάλει ένα μήνυμα καλωσορίσματος και ένα μικρό λογότυπο (deltaPC). Όπως αντιλαμβάνεστε, αυτή η εικόνα δεν παράγεται δυναμικά από το κύκλωμα γραφικών. Τη φτιάξαμε σ' ένα σχεδιαστικό πρόγραμμα του "μεγάλου" υπολογιστή και την ενσωματώσαμε στο firmware της GPU. Για την ακρίβεια, τα δεδομένα της εικόνας αποθηκεύτηκαν σε έναν πίνακα, ενώ προσθέσαμε στο πρόγραμμα και μια ρουτίνα που αντιγράφει το περιεχόμενο του συγκεκριμένου πίνακα στο "κέντρο" της frame buffer. Όλα αυτά είναι απλά και κατανοητά, αλλά κρύβουν ένα ενδιαφέρον ζήτημα: Το δικό μας κύκλωμα γραφικών χρησιμοποιεί ένα color format που δεν υπάρχει στα σύγχρονα σχεδιαστικά προγράμματα. Πώς πραγματοποιήσαμε την απαραίτητη μετατροπή;

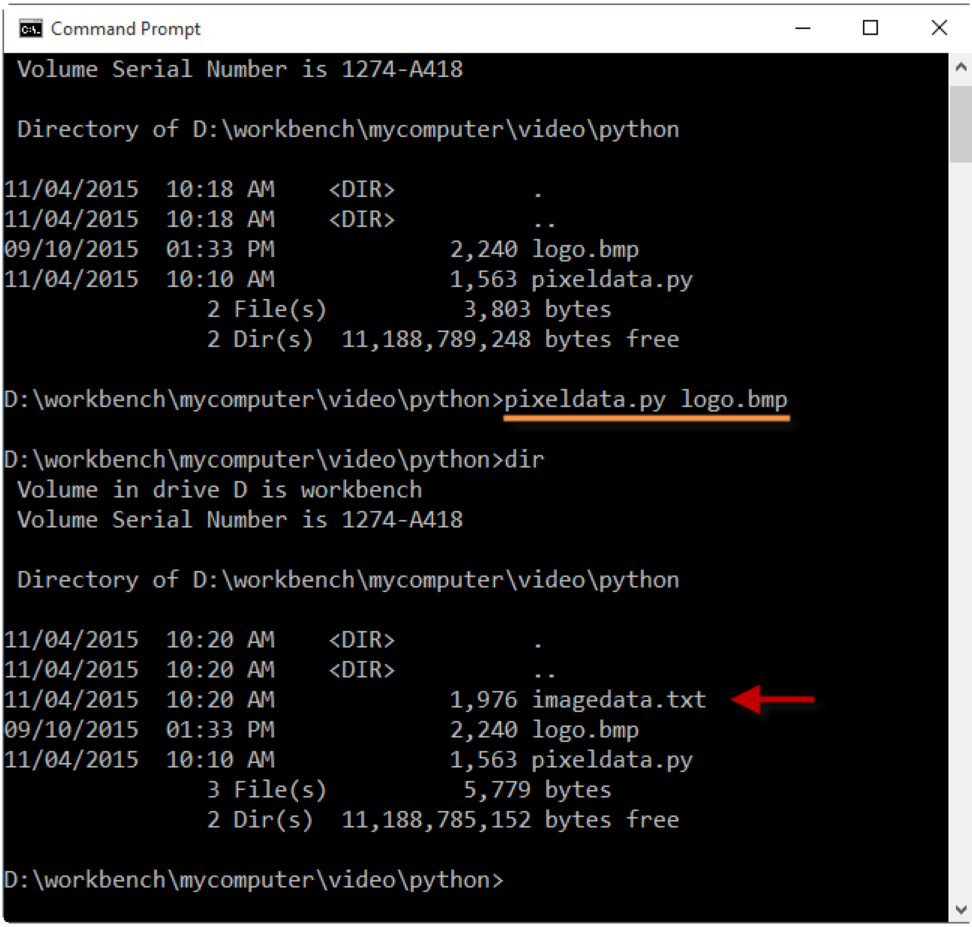


Το λογότυπο του υπολογιστή μας, όπως φαίνεται στο πρόγραμμα ζωγραφικής των Windows. Παρατηρείστε τη ρύθμιση της μεγέθυνσης που έχει τεθεί στο 800%!

Όταν σχεδιάσαμε το λογότυπο, το αποθηκεύσαμε σαν BMP με βάθος χρώματος 24bit. Επειδή οι εικόνες αυτού του τύπου δεν περιλαμβάνουν διαφάνειες (απουσιάζει το λεγόμενο alpha channel), τα 24 bits κατανέμονται αποκλειστικά στις τρεις χρωματικές συνιστώσες. Συνεπώς, για κάθε pixel του λογότυπου το αρχείο διέθετε 8 bits για το κόκκινο, 8 bits για το πράσινο και άλλα τόσα για το μπλε. Αυτό ήταν αρκετά βολικό, αλλά δεν έλυνε το πρόβλημα. Στο δικό μας κύκλωμα γραφικών, κάθε pixel περιγράφεται μόνο από 8 bits. Όπως θα θυμόσαστε, το πρώτο ζεύγος bits χρησιμοποιείται για το κόκκινο, το επόμενο ζεύγος για το πράσινο και το τρίτο για το μπλε. Τα τελευταία δύο bits ρυθμίζουν την ένταση όλων των χρωματικών συνιστωσών κι ως εκ τούτου περιγράφουν τη φωτεινότητα. Δεν ξέρουμε αν το συνειδητοποιείτε, αλλά η μετατροπή μεταξύ των συγκεκριμένων color formats είναι απλούστατη. Αφενός, τα δύο bits που προβλέπει το δικό μας πρότυπο για κάθε χρωματική συνιστώσα προκύπτουν ευθέως από τα 2 περισσότερο σημαντικά bits των αντίστοιχων οκτάδων του BMP. Αφετέρου, τα δύο bits φωτεινότητας του δικού μας προτύπου προκύπτουν από τα 2 περισσότερο σημαντικά bits της φωτεινότητας στο BMP. Αναρωτιέστε πού θα βρούμε τη φωτεινότητα των pixels μέσα στο BMP; Η αλήθεια είναι ότι δεν θα τη βρούμε πουθενά, αλλά μπορούμε να την υπολογίσουμε πανεύκολα. Αρκεί να σας πούμε ότι προκύπτει από το μέσο όρο των τιμών των τριών χρωμάτων. Τελικά, η διαδικασία της μετατροπής περιλαμβάνει μερικές απλές πράξεις, οι οποίες ωστόσο θα πρέπει να επαναληφθούν για κάθε pixel. Κάπως έτσι, επειδή απεχθανόμαστε τις χαμαλοδουλειές, βρήκαμε μια πρώτης τάξεως ευκαιρία για να καταφύγουμε στην αγαπημένη μας Python. Μπορείτε να κατεβάσετε το πρόγραμμα που γράψαμε από το http://bit.ly/pixeldata. Για την εύκολη ανάγνωση των δεδομένων της εικόνας χρησιμοποιήσαμε μια κλάση της βιβλιοθήκης PIL (Python Image Library). Σημειώστε, λοιπόν, ότι το πρόγραμμά μας προϋποθέτει την ύπαρξη της συγκεκριμένης βιβλιοθήκης. Ο τρόπος χρήσης του είναι εξαιρετικά απλός. Για κάθε εικόνα που του δίνουμε ως παράμετρο, δημιουργεί έναν πίνακα που μπορούμε να ενσωματώσουμε απευθείας στο firmware της GPU. Από εκεί και πέρα, αναλαμβάνει το πρόγραμμα και συγκεκριμένα μια ρουτίνα που αντιγράφει τα περιεχόμενα του πίνακα στη frame buffer.

Το αλατοπίπερο

Κάπου εδώ θα μπορούσαμε να σταματήσουμε. Παρουσιάσαμε όλα τα τμήματα του κώδικα κι έχετε μια πλήρη εικόνα της διάρθρωσης και της λειτουργίας του. Ωστόσο, αν η παρουσίαση τελείωνε εδώ, θα αποκομίζατε μια λανθασμένη αντίληψη για την ανάπτυξη του προγράμματος. Στην πορεία συναντήσαμε πάρα πολλά προβλήματα, τα περισσότερα από τα οποία δεν είχαν καμία σχέση με τις ενδιαφέρουσες προκλήσεις που μελετήσαμε. Αναφερόμαστε σε εκνευριστικά bugs που δοκίμασαν την υπομονή μας και σχετίζονταν αποκλειστικά με τη χρήση της γλώσσας Assembly. Για του λόγου το αληθές, στη συνέχεια θα παρουσιάσουμε το πιο ύπουλο σφάλμα στον κώδικα.



```
Command Prompt
 Volume Serial Number is 1274-A418

 Directory of D:\workbench\mycomputer\video\python

11/04/2015  10:18 AM    <DIR>          .
11/04/2015  10:18 AM    <DIR>          ..
09/10/2015  01:33 PM             2,240 logo.bmp
11/04/2015  10:10 AM             1,563 pixeldata.py
               2 File(s)          3,803 bytes
               2 Dir(s)  11,188,789,248 bytes free

D:\workbench\mycomputer\video\python>pixeldata.py logo.bmp

D:\workbench\mycomputer\video\python>dir
 Volume in drive D is workbench
 Volume Serial Number is 1274-A418

 Directory of D:\workbench\mycomputer\video\python

11/04/2015  10:20 AM    <DIR>          .
11/04/2015  10:20 AM    <DIR>          ..
11/04/2015  10:20 AM             1,976 imagedata.txt
09/10/2015  01:33 PM             2,240 logo.bmp
11/04/2015  10:10 AM             1,563 pixeldata.py
               3 File(s)          5,779 bytes
               2 Dir(s)  11,188,785,152 bytes free

D:\workbench\mycomputer\video\python>
```

Το προγραμματάκι που γράψαμε σε Python δέχεται σαν παράμετρο την εικόνα που θέλουμε να μετατρέψουμε και παράγει ένα αρχείο απλού κειμένου. Εκεί μέσα τοποθετείται ο πίνακας που πρέπει να ενσωματώσουμε στο firmware της GPU, για την προβολή της εικόνας από το κύκλωμα γραφικών.

Όταν η οθόνη άρχισε να αναγνωρίζει το σήμα που παρήγαγε το πρόγραμμά μας, ξεκινήσαμε μια σειρά ενθουσιωδών δοκιμών. Βέβαια, σε εκείνη τη φάση δεν είχαμε υλοποιήσει τις ρουτίνες επικοινωνίας και το κύκλωμα γραφικών δεν μπορούσε να δεχτεί εντολές από το περιβάλλον του. Έτσι, τα δοκιμαστικά μηνύματα που τυπώναμε στην οθόνη –σε διάφορες θέσεις και με ποικίλα χρώματα–, δημιουργούνταν με πρωτοβουλία της ίδιας της GPU. Αν κι επιβεβλημένος, αυτός ο τρόπος δοκιμής του σήματος ήταν ο πλέον κατάλληλος, αφού απομόνωνε όλους τους εξωτερικούς παράγοντες (προβλήματα στην επικοινωνία, σφάλματα στον κώδικα άλλων συστημάτων κ.ά.). Δυστυχώς όμως ο αρχικός ενθουσιασμός δεν κράτησε πολύ. Στη δεύ-

τερη ή τρίτη δοκιμή ήρθαμε αντιμέτωποι με ένα μυστήριο bug: Μερικές γραμμές pixels, λίγων μόνο χαρακτήρων, εκτείνονταν σε όλη τη γραμμή της οθόνης και είχαν απρόβλεπτο περιεχόμενο. Προσπαθώντας να συσχετίσουμε αυτό το παράξενο bug με κάποιο γνωστό μέγεθος, στην επόμενη έκδοση του κώδικα προσθέσαμε περισσότερα δοκιμαστικά μηνύματα, σε διάφορες θέσεις της οθόνης. Το αποτέλεσμα ήταν άκρως απογοητευτικό! Εκτός από το προηγούμενο bug εμφανίστηκε κι ένα νέο: Κάτω από μερικά γράμματα και ψηφία, τυπώνονταν όσοι χαρακτήρες βρίσκονται στις επόμενες θέσεις του πίνακα ASCII. Για παράδειγμα, κάτω από μια εμφάνιση του ψηφίου 9 (όχι όλες!) εμφανίζονταν οι χαρακτήρες :, ;, <, =, >, ?, @, A, B, C κ.ο.κ., μέχρι το z!

Τα δύο προβλήματα παρέπεμπαν στην ρουτίνα εμφάνισης των χαρακτήρων. Ήταν φανερό ότι κάποιοι μετρητές ξέφευγαν από τα προβλεπόμενα όρια και ο αντίστοιχος βρόχος επαναλήψεων λειτουργούσε ανεξέλεγκτα. Μόνο κάτι τέτοιο θα μπορούσε να δικαιολογήσει την προβολή περισσότερων pixels ή χαρακτήρων. Περιττό να πούμε ότι ακολούθησε μια εξονυχιστική διερεύνηση του κώδικα, η οποία ωστόσο αποδείχθηκε άκαρπη. Οι έλεγχοι στις τιμές των διαφόρων μετρητών (boundary checks) πραγματοποιούνταν ολόσωστα και δεν υπήρχε τίποτα που να δικαιολογεί την αλλοπρόσαλλη συμπεριφορά των βρόχων. Με τα πολλά, καταλήξαμε να ψάχνουμε σε μια φαινομενικά άσχετη θέση: Στον κώδικα της ρουτίνας scanline. Όπως ξέρουμε, η συγκεκριμένη ρουτίνα καλείται με ένα interrupt κι ορισμένες φορές η εκτέλεσή της συνέπιπτε με κάποιον έλεγχο. Προσέξτε τώρα το εξής: Στην Assembly, ακόμα και οι απλούστεροι έλεγχοι πραγματοποιούνται σε δύο βήματα. Για παράδειγμα, για να τσεκάρουμε αν ένας καταχωρητής έχει τιμή μεγαλύτερη ή μικρότερη του 47, κάνουμε τα εξής: Αφαιρούμε την τιμή 47 από την τιμή του καταχωρητή και ακολούθως ελέγχουμε αν το αποτέλεσμα της πράξης ήταν αρνητικό ή θετικό. Ωραία, θα πείτε, και πώς ελέγχουμε το αποτέλεσμα της αφαίρεσης; Αυτός ο έλεγχος πραγματοποιείται πανεύκολα, τσεκάροντας ένα συγκεκριμένο bit του καταχωρητή SREG. Ο εν λόγω καταχωρητής αντικατοπτρίζει την κατάσταση του μικροελεγκτή και, μεταξύ άλλων, περιγράφει το αποτέλεσμα της πιο πρόσφατης πράξης. Κάποιες φορές, λοιπόν, η ρουτίνα scanline ξεκινούσε αμέσως μετά την δοκιμαστική αφαίρεση κι ακριβώς πριν από τον έλεγχο του SREG. Μήπως πηγαίνει το μυαλό σας πουθενά; Μέσα στη ρουτίνα scanline πραγματοποιούνται πολλές πράξεις και η τιμή του SREG μεταβάλλεται. Επομένως, όταν η ροή του προγράμματος επιστρέφει σε κάποιον έλεγχο που είχε κοπεί στη μέση, η τιμή του SREG έχει πλέον αλλοιωθεί και η έκβαση του ελέγχου είναι πρακτικά τυχαία. Αυτό το bug είναι αρκετά ύπουλο, αλλά δεν μας είχε ξεφύγει. Το δικό μας λάθος ήταν προϊόν μεγαλύτερης απροσεξίας!

Γράφοντας τη ρουτίνα scanline, δεν είχαμε παραβλέψει το ενδεχόμενο του να διακόψει κάποιον έλεγχο. Έτσι, είχαμε μεριμνήσει για την αποθήκευση του SREG και φυσικά για την επαναφορά του. Δείτε πώς ξεκινούσε η ρουτίνα scanline:

```
cbi PORTB, hsync           ; (2) start horizontal sync pulse
in isr_temp1, PORTB        ; (1) save control signals
sbr isr_temp1, (1<<hsync) ; (1) restore initial hsync state
mov r3, isr_temp1          ; (1)
in r10, SREG               ; (1) save SREG
```

Ο παλμός HSYNC βρίσκεται στην έναρξη της γραμμής κι ακριβώς γι' αυτό, ο κώδικας ξεκινά με την πρώτη φάση του παλμού: Μετάβαση από HIGH σε LOW. Αμέσως μετά αποθηκεύουμε την κατάσταση των ακροδεκτών του PORTB. Σε αυτούς τους ακροδέκτες, όμως, συγκαταλέγεται κι εκείνος στον οποίο δημιουργείται ο παλμός HSYNC. Επομένως, για να αποθηκεύσουμε την αρχική, αναλλοίωτη κατάσταση των ακροδεκτών, δημιουργούμε ένα αντίγραφο των τιμών τους (isr_temp1) κι εκεί φροντίζουμε να επαναφέρουμε τον ακροδέκτη του παλμού HSYNC στην πρότερη κατάσταση. Κατόπιν αποθηκεύουμε την τιμή του isr_temp1, όπως και αυτήν του SREG. Μπορείτε να δείτε το λάθος;

Στο απόσπασμα που μελετάμε, οι εντολές cbi και sbr πετυχαίνουν κάτι παραπλήσιο: Αλλάζουν την τιμή ενός bit, σε κάποιον καταχωρητή. Αυτή η ομοιότητα είναι παραπλανητική! Οι εντολές cbi και sbr λειτουργούν πολύ διαφορετικά. Η πρώτη επιδρά μόνο σε ένα bit (το θέτει σε LOW), ενώ η δεύτερη μπορεί να επιδράσει ταυτόχρονα σε πολλά (τα θέτει σε HIGH). Η δεύτερη πετυχαίνει το σκοπό της πραγματοποιώντας ένα λογικό OR, μεταξύ του καταχωρητή και της τιμής που έχουμε δηλώσει. Ως εκ τούτου, ενώ η εκτέλεση της εντολής cbi δεν τροποποιεί ποτέ τον καταχωρητή SREG, η εκτέλεση της sbr τον επηρεάζει πάντα. Τελικά, για να διορθώσουμε το bug μετατοπίσαμε μια γραμμή κώδικα:

```
cbi PORTB, hsync           ; (2) start horizontal sync pulse
in r10, SREG               ; (1) save SREG --- before execution of sbr!
in isr_temp1, PORTB        ; (1) save control signals
sbr isr_temp1, (1<<hsync) ; (1) restore initial hsync state
mov r3, isr_temp1          ; (1)
```

Χρειάζεται να πούμε κάτι περισσότερο; Η Assembly προσφέρει τον υπέρτατο έλεγχο οποιουδήποτε μικροελεγκτή κι ενώ αυτό μοιάζει με προτέρημα, μπορεί άνετα να μετατραπεί σε εφιάλτη. Σκεφτείτε ότι πατήσαμε πολλές μπανανόφλουδες σαν αυτή και κάθε φορά ταλαιπωρούμασταν για μέρες. Ευτυχώς που η κατασκευή ενός υπολογιστή αποτελεί μακράν τον πιο συναρπαστικό στόχο, γιατί θα είχαμε παραιτηθεί από πολύ νωρίς.



deltaHacker Field Bag
€25.50
MORE COLORS AVAILABLE
deltaHacker Mugs
€12.50
deltaHacker Thermos® Can Cooler
€17.50
QUESTIONS?
ORDER BY PHONE?
CALL 1-877-809-1659
Where's My Stuff?
Track your recent orders.
View or change your order.
Shipping & Returns
See our shipping rates.
Make a return or exchange.
Need Help?
Satisfaction Guaranteed.
Visit our Customer Service.
is powered by CafePress Inc. Copyright © 1999-2015 CafePress.com. All rights reserved.
olicy | Trademark & Copyright Information
urrency rates are updated daily and may fluctuate.
KER
Δώρα με στιλ, που έχουν *αληθινή* αξία.
Όλα στο μαγαζάκι του deltaHacker,
κάτω από το Cafepress.
Περάστε μια βόλτα από το
http://www.cafepress.co.uk/deltahacker
–σας περιμένουμε!

Skill: Intermediate
Tags: encryption, tunnelling, VPN, OpenVPN, OpenVPN Access Server, pfSense, outbound NAT

Ασφαλής δίοδος προς το Internet, για ολόκληρο το LAN

Η προστασία των συνδέσεών μας ακόμη κι όταν ξεκινούν από επισφαλή σημεία πρόσβασης, όπως, π.χ., είναι τα δημόσια WiFi hotspots, είναι ένα ζήτημα που μας απασχολεί συχνά. Μια καλή λύση είναι να βγαίνουμε στον έξω κόσμο μέσω ενός κρυπτογραφημένου καναλιού, το οποίο ξεκινά από τον υπολογιστή μας και καταλήγει σε έναν απομακρυσμένο VPN server. Τι γίνεται όμως αν θέλουμε να προστατεύσουμε όλες τις συνδέσεις, όλων των μηχανημάτων και συσκευών, ενός ολόκληρου τοπικού δικτύου;

του Χρήστου Βαρελά

Σε πρώτη φάση θα μπορούσαμε να σκεφτούμε ως εξής:

- στήνουμε και ρυθμίζουμε VPN server, σε κάποιο VPS στο cloud
- εγκαθιστούμε –και φυσικά ρυθμίζουμε– τον κατάλληλο VPN client σε κάθε υπολογιστή, laptop, smartphone, tablet που διαθέτουμε
- φροντίζουμε ώστε ο client κάθε συσκευής να τρέχει αυτόματα κατά την εκκίνηση/ενεργοποίησή της.

Έτσι, οι συνδέσεις όλων των συσκευών είναι κρυπτογραφημένες και προστατευμένες κι εμείς απόλυτα ευχαριστημένοι.

Ή μπορεί και να μην είμαστε.

Η παραπάνω λύση, βλέπετε, είναι τουλάχιστον άκομψη, απαιτεί αρκετή δουλειά σε κάθε μία συσκευή, ενώ τελικά αυτό που κάνουμε είναι ναι μεν να τις προστατεύουμε ξεχωριστά, αλλά όχι *ολόκληρο* το τοπικό δίκτυο. Αν, δηλαδή, εισέλθει στο δίκτυο ένα νέο μέλος, τότε θα πρέπει να εγκαταστήσουμε και να ρυθμίσουμε και σ' αυτό τον απαραίτητο VPN client.

Ιδανικά, αυτό που θα θέλαμε θα ήταν οι συνδέσεις κάθε συσκευής *εντός* του τοπικού μας δικτύου να προστατεύονται *αυτόματα* και χωρίς την εγκατάσταση client software. Θα επιθυμούσαμε επιπρόσθετα τη δυνατότητα για κεντρικό καθορισμό εξαιρέσεων, είτε για συσκευές που για οποιονδήποτε λόγο δεν έχει νόημα να βγαίνουν στο Internet μέσω VPN, είτε για προορισμούς στους οποίους προτιμάμε να φτάνουμε *πάντα* μέσω του ISP μας κι όχι μέσω του απομακρυσμένου VPN server.

Είναι προφανές ότι το επιθυμητό αυτό σενάριο καθίσταται δυνατό αρκεί ο router του τοπικού δικτύου να έχει την ικανότητα να λειτουργεί ως client του VPN server. Πριν παρουσιάσουμε, λεπτομερώς και βήμα προς βήμα, τη λύση που εμείς υλοποιήσαμε, ας συζητήσουμε λίγο αναλυτικότερα το γιατί θα θέλαμε μια τέτοια διευθέτηση για το τοπικό μας δίκτυο.

Προς τι όλο αυτό το κακό;

Ο καθένας θα 'χει τους λόγους του για να κρυπτογραφεί τη δικτυακή του κίνηση, πιστεύουμε όμως ότι δύο είναι τα ισχυρότερα κίνητρα.

Ιδιωτικότητα. Ο ISP μας δεν πρέπει να έχει την παραμικρή ιδέα για το τι κάνουμε στο Internet. Μια αποτελεσματική λύση είναι η αναδρομολόγηση όλης της κίνησης του δικτύου μας μέσω ενός κρυπτογραφημένου τούνελ. Η κρυπτογράφηση των πακέτων γίνεται πριν αυτά εγκαταλείψουν τον router μας, ενώ η αποκρυπτογράφησή τους πραγματοποιείται αφού φτάσουν στον απομακρυσμένο VPN server. Ο ISP βρίσκεται κάπου ανάμεσα και δεν έχει καμία, μα καμία δυνατότητα για αποκρυπτογράφηση. Το ποια sites επισκεπτόμαστε, το ποιες δικτυακές υπηρεσίες χρησιμοποιούμε, τα DNS queries που αποστέλλουμε κ.ο.κ., ξαφνικά αποτελούν μέγιστα μυστήρια για τον ISP. Το ωραίο της υπόθεσης είναι πως δεν μπορεί να κάνει κάτι επ' αυτού.

Πρόσβαση σε υπηρεσίες με γεωγραφικούς περιορισμούς. Αναλόγως της θέσης που βρίσκεται ο VPN server, καθίσταται δυνατό ν' αποκτήσουμε πρόσβαση σε υπηρεσίες στις οποίες προηγουμένως η πρόσβαση απαγορευόταν. Παράδειγμα: Στο πλήρες περιεχόμενο του Hulu έχουν πρόσβαση μόνον Αμερικανοί. Αν όμως ο VPN server λειτουργεί σε VPS που βρίσκεται σε αμερικανικό data center, τότε για το Internet το δημόσιο IP μας είναι επίσης αμερικανικό κι επομένως αποκτάμε κι εμείς πλήρη πρόσβαση στο Hulu.

Κατά τη δική μας τουλάχιστον άποψη, τα δύο προαναφερθέντα κίνητρα αρκούν ώστε κάποιος να φροντίσει για VPN tunnelling σε επίπεδο τοπικού δικτύου. Υπάρχουν όμως και κάποια "αλλά", τα οποία οφείλουμε να γνωρίζουμε ευθύς εξαρχής.

Ταχύτητα. Αναλόγως του πού βρίσκεται ο VPN server, είναι πιθανό να διαπιστώσουμε ότι η πρόσβαση σε συγκεκριμένες υπηρεσίες και sites γίνεται πιο αργά σε σύγκριση με την πρότερη κατάσταση, κατά την οποία βγαίναμε στο Internet απευθείας κι όχι μέσω VPN. Το ευτύχημα είναι ότι μπορούμε να ορίζουμε εξαιρέσεις, στο επίπεδο του router. Παράδειγμα: Αν η πρόσβαση σε ένα σύστημα e-banking ξαφνικά έγινε αργή, αφού τώρα δρομολογείται μέσω data center στη Νέα Υόρκη, τότε είναι εύκολο να ορίσουμε έναν κανόνα που θα λέει ότι η πρόσβαση στην τράπεζα *δεν* θα γίνεται μέσω VPN αλλά απευθείας, δηλαδή μέσω του ISP μας. Άλλο παράδειγμα: Αν υπάρχει ένα PC το οποίο χρησιμοποιούμε κυρίως για online gaming, κάλλιστα μπορούμε να ορίσουμε κανόνα στον router που θα λέει ότι το συγκεκριμένο PC θα βγαίνει απευθείας στο Internet, παρακάμπτοντας το VPN tunnel.

Blacklisting. Για υπηρεσίες που παρέχονται με γεωγραφικούς περιορισμούς είναι πιθανό να τηρούνται λίστες με διευθύνσεις IP παρόχων, μέσω των οποίων έχει διαπιστωθεί ότι επιχειρείται "πλάγια" πρόσβαση. Σε μια τέτοια περίπτωση, ακόμη κι αν βγαίνουμε στο Internet, π.χ., με αμερικανικό IP, δεν θα έχουμε πρόσβαση σε όλες τις

επιθυμητές υπηρεσίες. Αν λοιπόν θέλετε VPN μόνο και μόνο για να έχετε πρόσβαση σε υπηρεσίες με γεωγραφικούς περιορισμούς, κάντε πρώτα μια σχετική έρευνα. Πιθανώς να δείτε ότι πρέπει να στήσετε τον VPN server σε συγκεκριμένο data center, συγκεκριμένου VPS provider. Ενδέχεται επίσης να διαπιστώσετε ότι δεν έχει νόημα να στήσετε τον δικό σας VPN server, μόνος σας. Αντίθετα, ίσως αξίζει να καταφύγετε στις υπηρεσίες ενός VPN provider. Επιλέξτε κάποιον ο οποίος αφενός σέβεται την ιδιωτικότητα των πελατών του, αφετέρου έχει τη δυνατότητα για δυναμική αλλαγή VPN end-point ώστε ο εκάστοτε πελάτης να επιλέγει ανά πάσα στιγμή από ποια γεωγραφική περιοχή θα αναδύεται στο δημόσιο Intetnet.

Περιοριστική νομοθεσία. Όπως αναφέραμε και πριν λίγο, όταν χρησιμοποιούμε VPN τότε ο ISP εξακολουθεί βεβαίως να βλέπει δικτυακή κίνηση από μέρους μας. Τα πακέτα όμως είναι κρυπτογραφημένα, ο ISP δεν μπορεί να τα αποκρυπτογραφήσει και, επιπρόσθετα, δεν έχει δικαίωμα να κάνει κάτι επ' αυτού, αφού "packets are packets and there ain't nothin' you can do about it". (Φαντάζεστε να υπάρχει υπάλληλος σε ελληνικό ISP, ο οποίος χρησιμοποιεί αυτή τη φράση; Αν υπάρχει και διαβάζει το παρόν ας επικοινωνήσει μαζί μας, θα θέλαμε να τον γνωρίσουμε!) Μολαταύτα, έχει επιχειρηθεί να περάσει νομοθεσία που θα επιτρέπει ή ακόμη και θα υποχρεώνει τους ISPs να μειώνουν δραματικά την ταχύτητα σύνδεσης των πελατών που χρησιμοποιούν VPN. Το σκεπτικό είναι ότι μόνον όσοι διακινούν υλικό που προστατεύεται από πνευματικά δικαιώματα έχουν λόγο να χρησιμοποιούν υπηρεσίες VPN, οπότε χτύπησέ τους εκεί που τους πονάει. Δεν γνωρίζουμε να ισχύει (ακόμα) σχετική νομοθεσία στην Ευρώπη, βεβαίως ούτε και στη χώρα μας. Επειδή όμως ποτέ δεν ξέρουμε τι μας ξημερώνει (http://deltahacker.gr/?p=15032), ας έχουμε στην άκρη του μυαλού μας κι αυτή την πιθανότητα.

Ως αναγνώστες του deltaHacker γνωρίζουμε ότι δεν αποθαρρύνεστε εύκολα από ζητήματα σαν αυτά που μόλις αναφέραμε. Ας συνεχίσουμε λοιπόν με τα προαπαιτούμενα για την κρυπτογράφηση της δικτυακής κίνησης *ολόκληρου* του τοπικού μας δικτύου. Ίσως χρειαστεί αρκετή δουλειά κατά την προετοιμασία, πιστεύουμε ωστόσο ότι αξίζει τον κόπο και το αποτέλεσμα θα σας ανταμείψει.

Τα προαπαιτούμενα

Κατ' αρχάς χρειαζόμαστε έναν OpenVPN Access Server στο cloud. Θα μπορούσαμε να δουλέψουμε και με την εκδοχή Community Edition του OpenVPN, όμως τα πράγματα είναι γενικά ευκολότερα με την εκδοχή Access Server. Σε περίπτωση που *ακόμη* δεν έχετε τον δικό σας OpenVPN Access Server (δεν πιστεύουμε να μην έχετε αλλά λέμε τώρα), διαβάστε το αναλυτικό άρθρο που ξεκινά από τη σελίδα 14 του παρόντος τεύχους.

Για το τοπικό μας δίκτυο (LAN) χρειαζόμαστε έναν router ο οποίος μπορεί να λειτουργεί κι ως OpenVPN client. Υποψήφιοι είναι όλοι οι routers που δέχονται το dd-wrt (http://deltahacker.gr/?p=6208) ή το OpenWrt (http://deltahacker.gr/?p=7330). Εναλλακτικά, σε ρόλο router για το LAN ίσως έχουμε ένα μηχανάκι το οποίο τρέχει Linux ή *BSD, αφού κι αυτό είναι σε θέση να λειτουργεί ως OpenVPN client. Υπάρχει

επίσης και η οδός του pfSense, το οποίο είναι το αγαπημένο μας λετουργικό σύστημα για χρήση router/firewall και τυχαίνει μάλιστα να το χρησιμοποιούμε στα κεντρικά της Parabing Creations, στο Northern Outpost (του SKG).

Στο παρόν άρθρο δουλεύουμε με το pfSense. Αν δεν το χρησιμοποιείτε ήδη ή για κάποιον ανεξήγητο λόγο διστάζετε να το εγκαταστήσετε, ίσως πρέπει να παρακολουθήσετε το 7ο επεισόδιο της 1ης σεζόν του deltaCast: http://deltahacker.gr/deltacast-s01e07. Το επεισόδιο είναι λίγο παλιό, όμως το σχετικό video tutorial είναι αναλυτικό κι εκτενές κι έχουμε την ελπίδα ότι *και* εσείς θα πωρωθείτε με το pfSense.

Δημιουργία νέου χρήστη & προφίλ σύνδεσης

Συνδεόμαστε μέσω SSH στο VPS όπου φιλοξενείται ο OpenVPN Access Server και δημιουργούμε ένα νέο χρήστη. Σε πολύ λίγο, για τον αντίστοιχο λογαριασμό ο Access Server θα μας φτιάξει κι ένα προφίλ σύνδεσης. Μπορείτε φυσικά να ονομάσετε το νέο σας χρήστη όπως θέλετε. Εμείς επιλέξαμε το χαρακτηριστικό όνομα pfsense και, όπως βλέπετε παρακάτω, ζητήσαμε από το εργαλείο adduser να μην του φτιάξει προσωπικό κατάλογο:

```
sub0@defiant:~$ sudo adduser --no-create-home pfsense
```

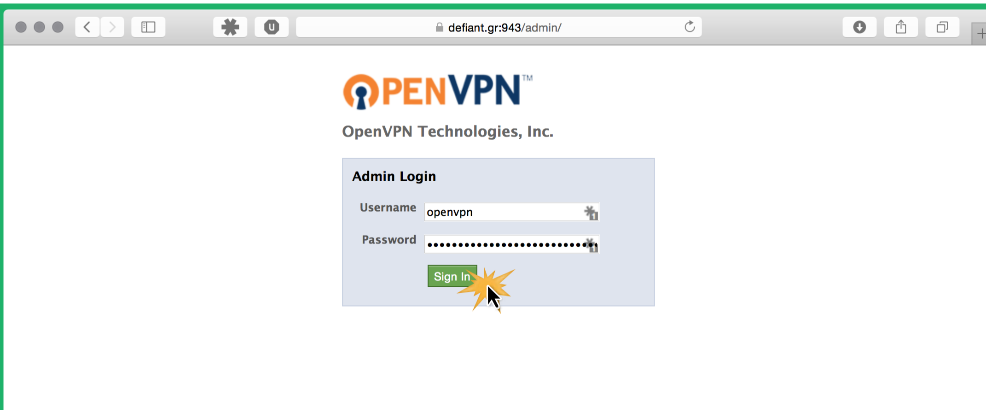
Δείτε σοβαρά το password για το νέο χρήστη και καθορίστε του ένα εξαιρετικά ισχυρό συνθηματικό.

cvar — sub0@defiant: ~ — ssh — 115×30 — ⌘1

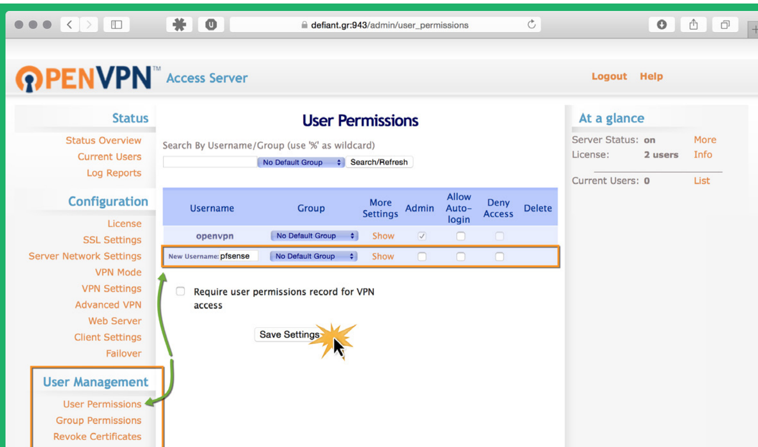
```
sub0@defiant:~$ sudo adduser --no-create-home pfsense
Adding user `pfsense' ...
Adding new group `pfsense' (1003) ...
Adding new user `pfsense' (1003) with group `pfsense' ...
Not creating home directory `/home/pfsense'.
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for pfsense
Enter the new value, or press ENTER for the default
        Full Name []: pfSense @ SKG
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
sub0@defiant:~$
```

Δημιουργία νέου χρήστη με username το pfsense και χωρίς προσωπικό κατάλογο. Για τον αντίστοιχο λογαριασμό, ο OpenVPN Access Server θα μας φτιάξει σε λίγο ένα προφίλ σύνδεσης. Σημαντικό: Μην παραλείψετε να δώσετε στο χρήστη ένα ισχυρό password!

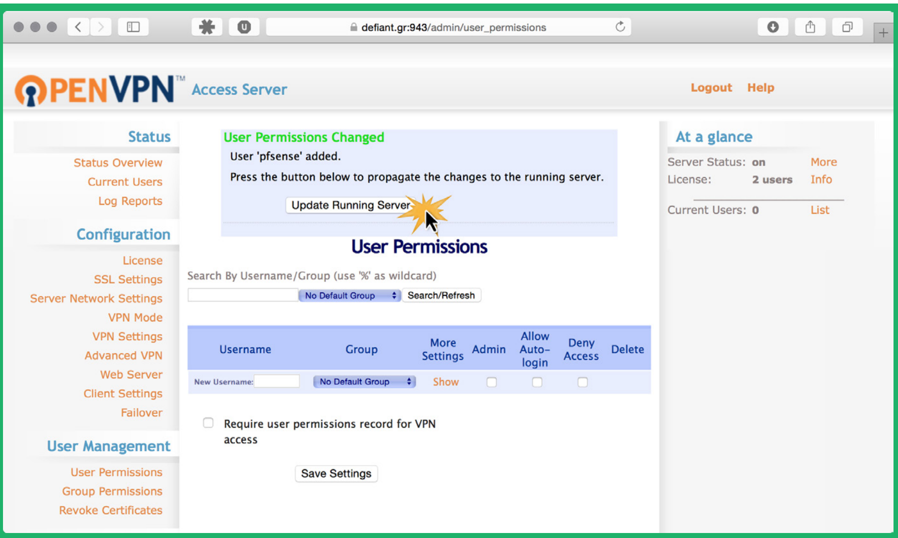
Τη δημιουργία του προφίλ σύνδεσης που μόλις αναφέραμε τη διευθετούμε μέσα από το web panel του OpenVPN Access Server. Δείτε τα ακόλουθα screen shots και διαβάστε τις αντίστοιχες περιγραφές.



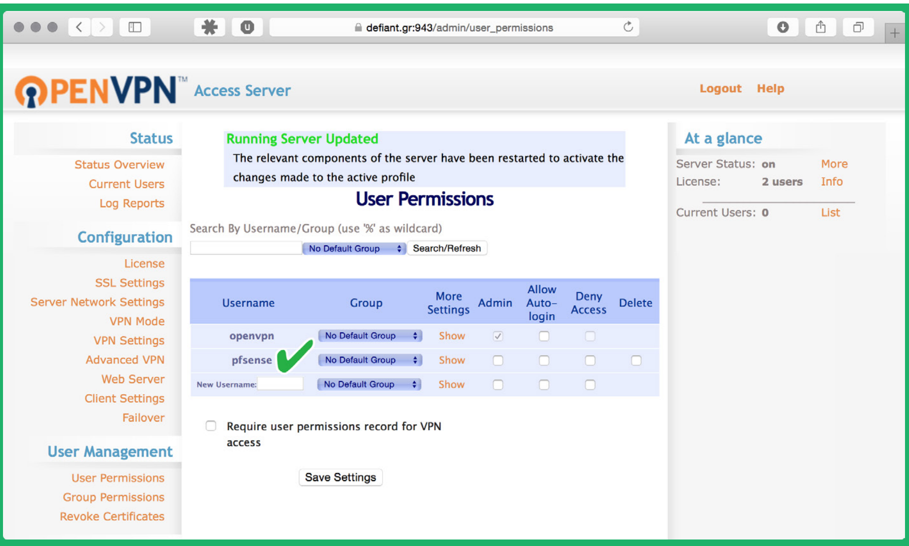
Από τον web browser της προτίμησής μας μεταβαίνουμε στη διεύθυνση **https://δημόσιο_IP_του_AS:943/amdin**. Για τον δικό μας Access Server, στη θέση του "δημόσιο_IP_του_AS" αρκεί να βάλουμε το "defiant.gr". Συνδεόμαστε στο λογαριασμό του διαχειριστή του Access Server, το username του οποίου εξ ορισμού είναι το openvpn.



Από την κατηγορία ρυθμίσεων User Management, στην αριστερή στήλη του web panel, επιλέγουμε το User Permissions. Στο New Username πληκτρολογούμε το όνομα του χρήστη που φτιάξαμε πριν λίγο με το προγραμματάκι adduser, από τη γραμμή εντολών του VPS. Ο δικός μας χρήστη ονομάζεμαι pfsense. Φροντίζουμε ώστε να *μην* είναι τσεκαρισμένο κανένα από τα Admin, Allow Auto-login και Deny Access στα δεξιά. Επικυρώνουμε με κλικ στο κουμπί Save Settings.



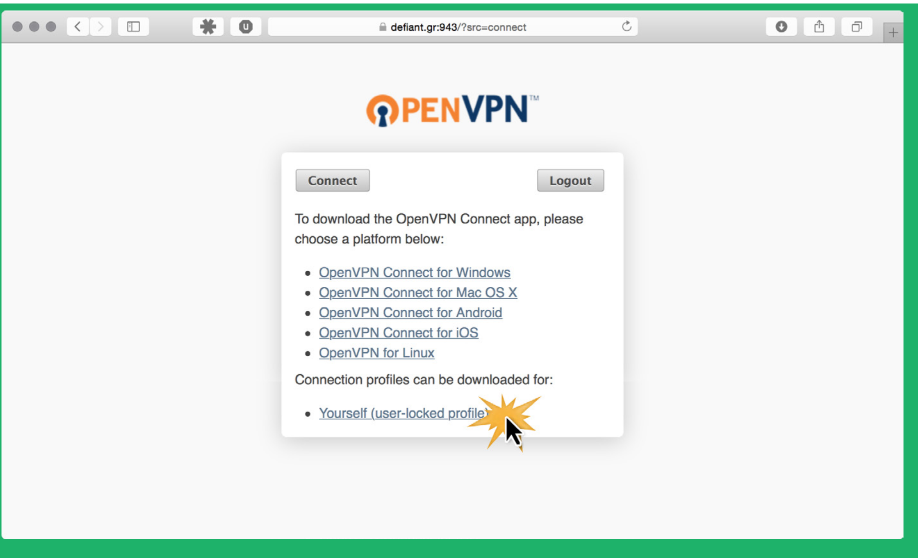
Και για τον Access Server υπάρχει, πλέον, άλλος ένας χρήστης, ονόματι pfsense. Προκειμένου να ενημερωθεί ο server για την παρουσία του, αρκεί ένα κλικ στο κουμπί Update Running Server.



Ιδού οι δύο χρήστες που γνωρίζει και διαχειρίζεται ο OpenVPN Access Server. Για περισσότερους χρήστες χρειάζεται να αγοράσουμε κατάλληλο license. Οι δύο υπάρχοντες, πάντως, είναι ό,τι πρέπει για τις ανάγκες μας, αφού για καθέναν εξ αυτών επιτρέπονται τα concurrent connections.



Αποσυνδεόμαστε από το web panel του OpenVPN Access Server κι αμέσως συνδεόμαστε στο **https://defiant.gr:943**, αλλά αυτή τη φορά στο λογαριασμό του χρήστη pfsense. Πριν κάνουμε κλικ στο OK φροντίζουμε ώστε στο drop-down menu αριστερά να 'ναι επιλεγμένο το Login.

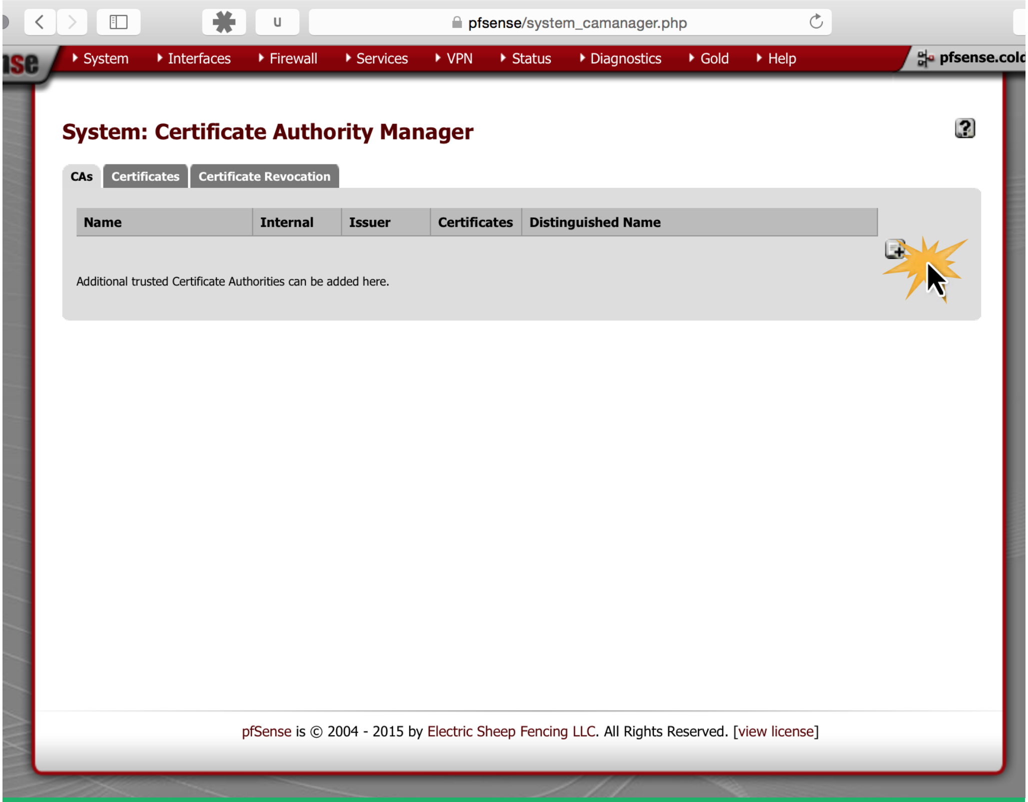


Από τη νέα σελίδα που εμφανίζεται παίρνουμε το προφίλ σύνδεσης για τον χρήστη ονόματι pfsense, με ένα κλικ στο σύνδεσμο Yourself (user-locked profile).

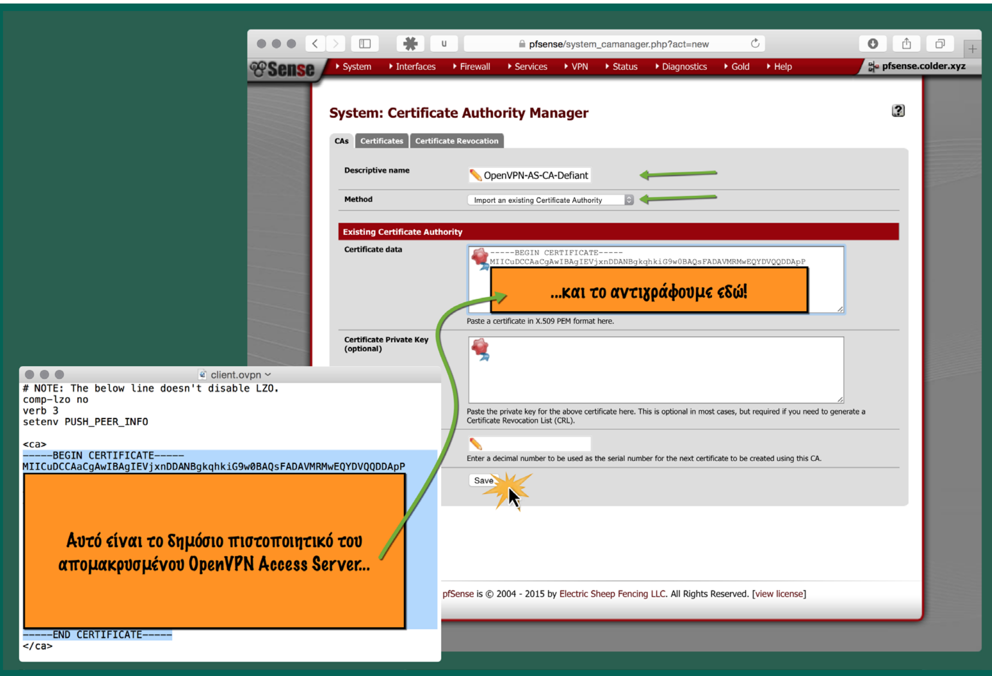
Αφού λοιπόν κατεβάσουμε στον υπολογιστή μας το προφίλ σύνδεσης για το λογαριασμό του χρήστη pfsense –πρόκειται για ένα αρχείο με όνομα client.ovpn– μπορούμε να αποσυνδεθούμε από το web panel του Access Server πατώντας στο Logout.

Εισαγωγή πιστοποιητικών & κλειδιών στο pfSense

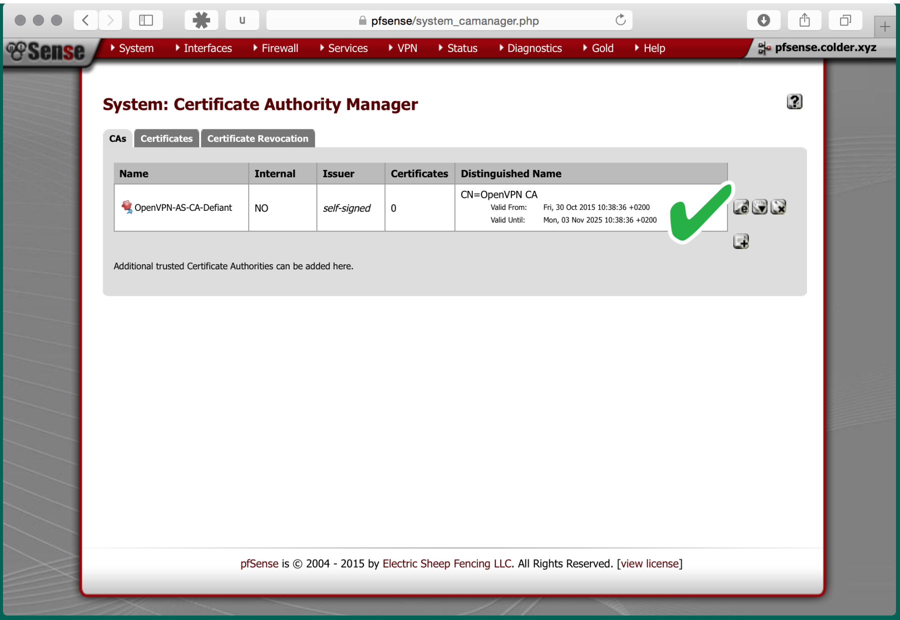
Το αρχειάκι client.ovpn το ανοίγουμε με έναν απλό text editor και το κρατάμε εύκαιρο στην επιφάνεια εργασίας. Κατόπιν, από έναν web browser συνδεόμαστε στο περιβάλλον διαχείρισης του pfSense-based router μας. Δείτε στα ακόλουθα screen shots πώς πληροφορούμε το pfSense για το δημόσιο πιστοποιητικό του απομακρυσμένου OpenVPN Access Server, καθώς και για το πιστοποιητικό αλλά και για το ιδιωτικό κλειδί του πελάτη που μας ενδιαφέρει (είναι εκείνος με username το pfsense).



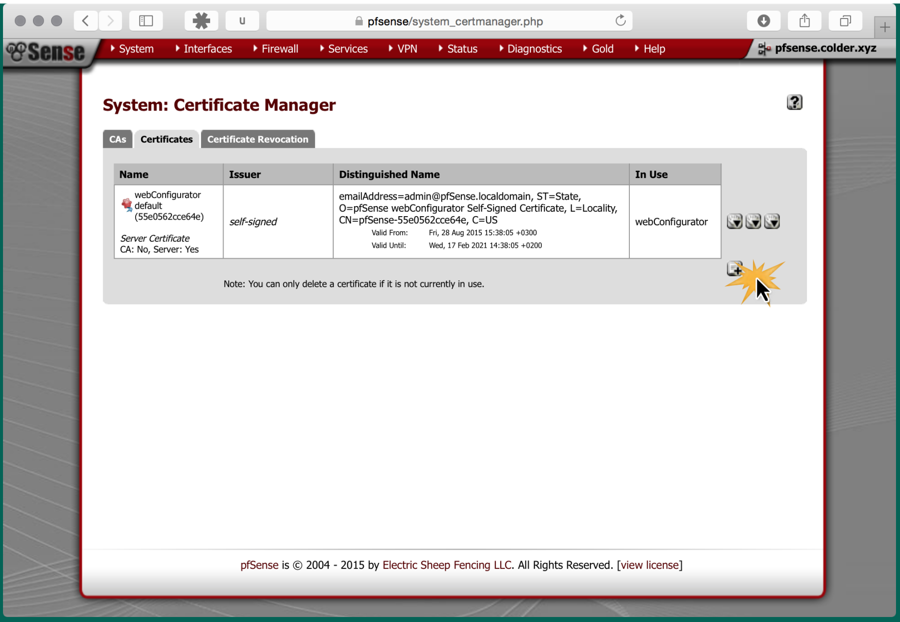
Στο περιβάλλον του pfSense ακολουθούμε τη διαδρομή System → Cert Manager. Από τις τρεις καρτέλες της σελίδας "System: Certificate Authority Manager" φροντίζουμε ώστε να είναι επιλεγμένη η αριστερότερη, δηλαδή η CAs. Κάνουμε ένα κλικ στο εικονίδιο με το "συν", στα δεξιά.



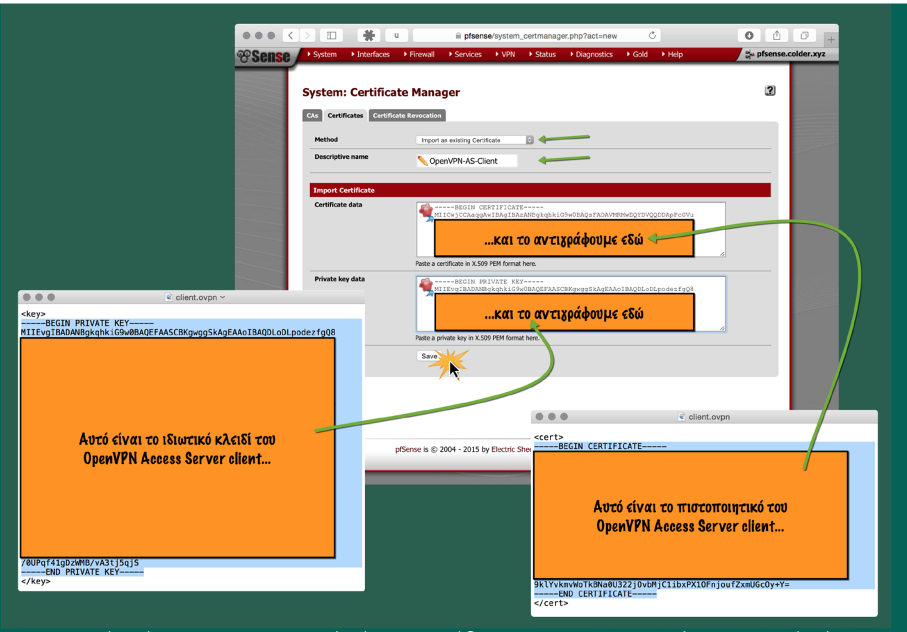
Θα δώσουμε στο pfSense το δημόσιο κλειδί του απομακρυσμένου OpenVPN Access Server. Στο πεδίο Descriptive name πληκτρολογούμε κάτι περιγραφικό, ώστε με μια ματιά να καταλαβαίνουμε σε τι αφορά το πιστοποιητικό. Εμείς, π.χ., σ' αυτή τη θυρίδα γράψαμε OpenVPN-AS-CA-Defiant. Ακριβώς από κάτω, για το Method επιλέγουμε το Import an existing Certificate Authority. Στην περιοχή Existing Certificate Authority –και συγκεκριμένα στη θυρίδα ονόματι Certificate data–, αντιγράφουμε το πιστοποιητικό του Access Server: Βρίσκεται στο αρχείο client.ovpn και συγκεκριμένα ανάμεσα στις ετικέτες <ca> και </ca>. Προσοχή: Αντιγράφουμε τα πάντα μεταξύ των δύο αυτών ετικετών, συμπεριλαμβανομένων των -----BEGIN CERTIFICATE----- και -----END CERTIFICATE-----, αλλά όχι τις ίδιες τις ετικέτες. Αγνοούμε το πλαίσιο ονόματι Certificate Private key (optional), από κάτω. Επικυρώνουμε την εισαγωγή με κλικ στο κουμπάκι Save.



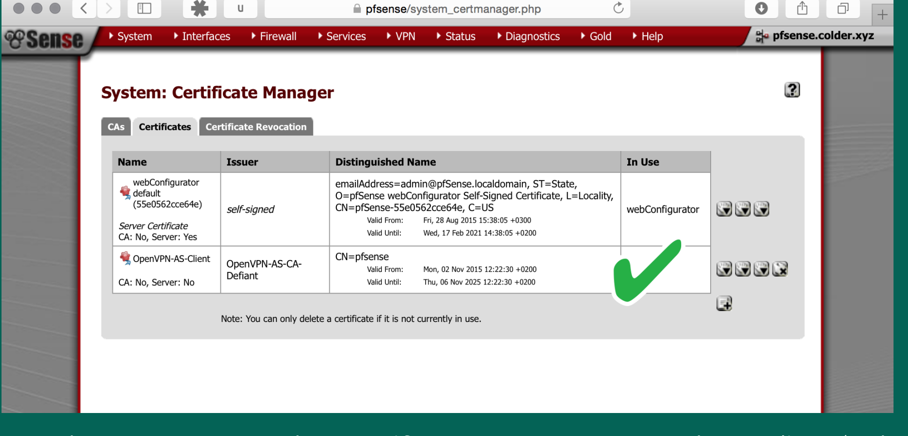
Το pfSense γνωρίζει, πλέον, το δημόσιο πιστοποιητικό του απομακρυσμένου OpenVPN Access Server.



Πηγαίνουμε στην καρτέλα Certificates, στα δεξιά της CAs. Εδώ θα εισαγάγουμε το πιστοποιητικό καθώς και το ιδιωτικό κλειδί του πελάτη (με username το pfsense), οπότε ξεκινάμε με ένα κλικ στο κουμπάκι "συν", κάτω δεξιά.



Θέτουμε Method = Import an existing Certificate και στη θυρίδα Descriptive name, από κάτω, πληκτρολογούμε μια χαρακτηριστική περιγραφή (π.χ., "OpenVPN-AS-Client"). Το πιστοποιητικό του πελάτη είναι στο αρχείο client.ovpn, ανάμετα στις ετικέτες <cert> και </cert>, και το αντιγράφουμε στο πλαίσιο Certificate data. Το δε ιδιωτικό κλειδί του πελάτη είναι επίσης στο αρχείο client.ovpn, ανάμεσα στις ετικέτες <key> και </key>, κι αυτό το μεταφέρουμε στο πλαίσιο Private key data. Επικυρώνουμε τις δύο αυτές εισαγωγές με κλικ στο κουμπί Save.

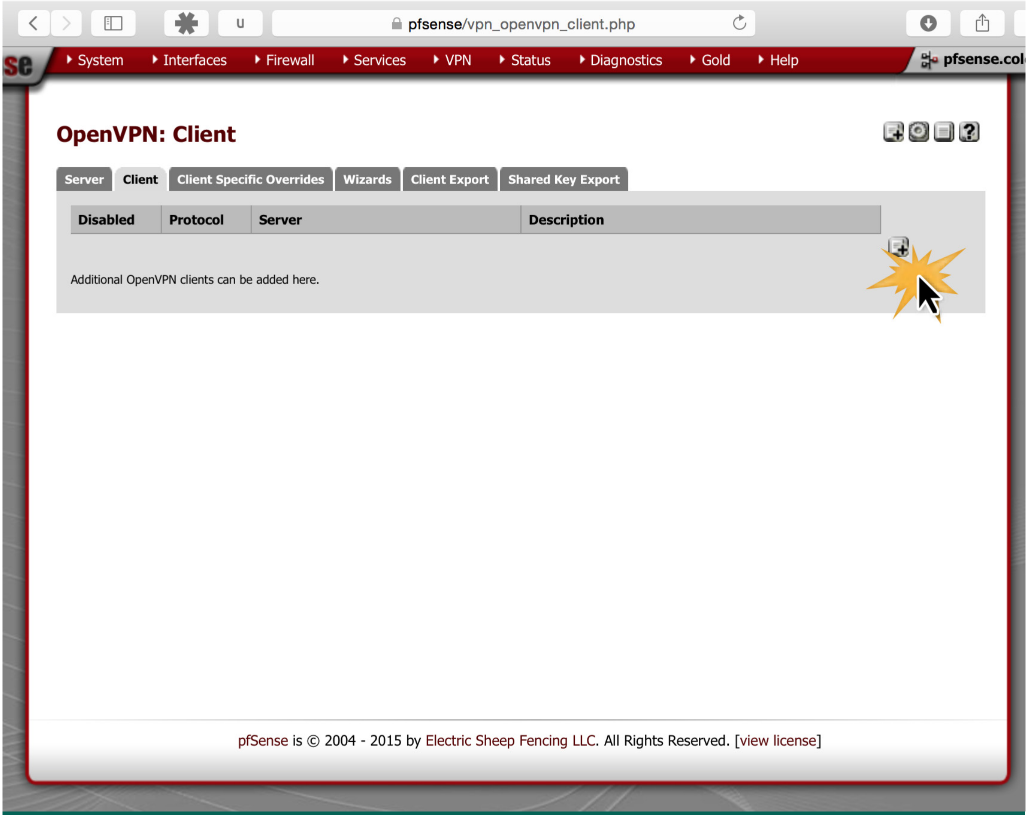


Όπως βλέπουμε στην καρτέλα Certificates, το πιστοποιητικό του client (όνοματι pfsense) έχει υπογραφεί από το πιστοποιητικό του OpenVPN Access Server κι έχει εισαχθεί επιτυχώς. Τέλεια όλα.

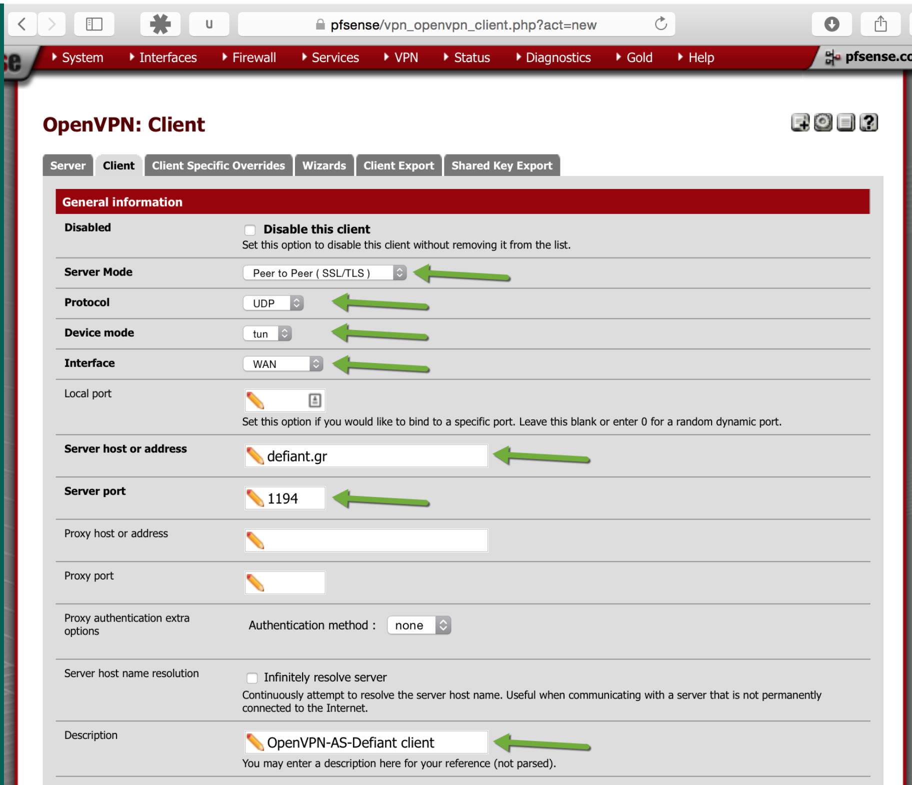
Μπορούμε τώρα να προχωρήσουμε στη ρύθμιση του pfSense, ώστε να λειτουργεί ως client του OpenVPN Access Server.

Ρύθμιση του OpenVPN client, στο pfSense

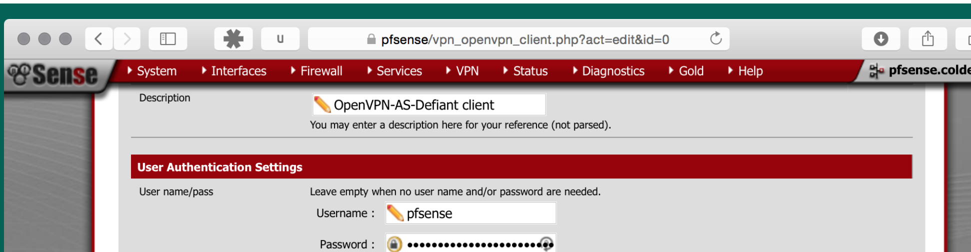
Παραμένοντας στο περιβάλλον διαχείρισης του pfSense-based router μας, επιλέγουμε VPN → OpenVPN → Client. Όλες τις λεπτομέρειες για τη ρύθμιση του client θα τις δείτε στα screen shots που ακολουθούν και φυσικά θα τις διαβάστε στις αντίστοιχες περιγραφές.



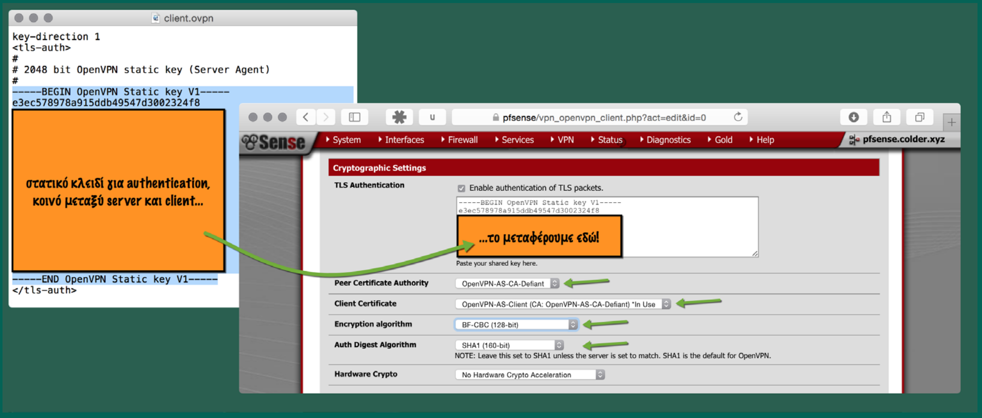
Στη σελίδα με όνομα "OpenVPN: Client" κάνουμε ένα κλικ στο κουμπάκι "συν", κάτω δεξιά, ώστε να προσθέσουμε ένα νέο client instance.



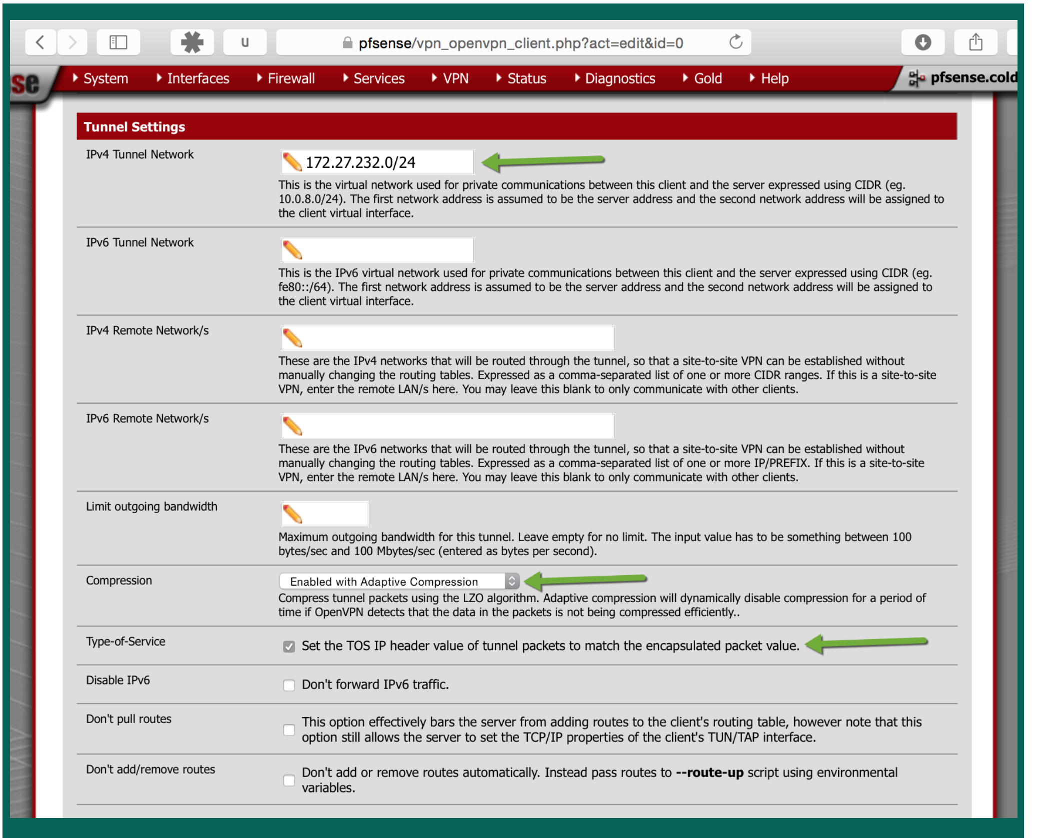
Για τις πρώτες βασικές ρυθμίσεις, οι οποίες διευθετούνται στην ενότητα General information, φροντίστε ώστε να ισχύουν όλα όσα υποδεικνύονται από τα βελάκια. Βεβαίως, στη θυρίδα Server host or address θα πληκτρολογήσετε είτε το domain name του VPS σας είτε τη δημόσια διεύθυνση IP που έχει. Επιπρόσθετα, στο πεδίο Description είναι καλή ιδέα να γράψετε κάτι περιγραφικό για τον συγκεκριμένο client.



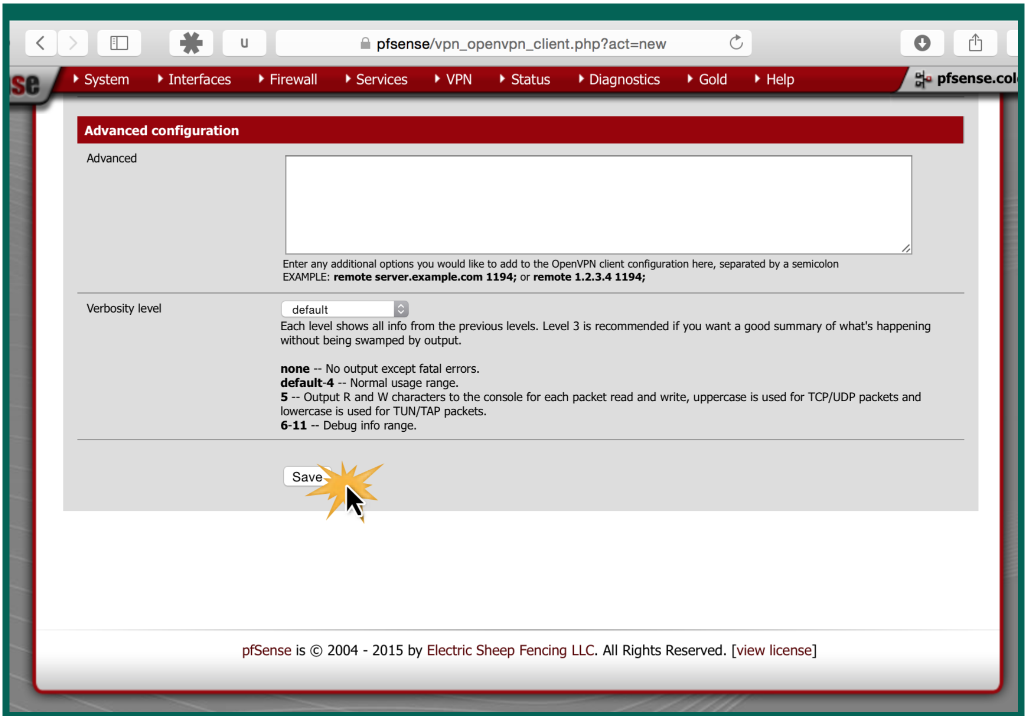
Παραμένουμε στην ίδια σελίδα και σκρολάρουμε λίγο πιο κάτω, έως ότου δούμε την ενότητα User Authentication Settings. Στη θυρίδα Username πληκτρολογούμε το όνομα του απομακρυσμένου χρήστη, στο λογαριασμό του οποίου θα συνδεθεί ο router (για το παράδειγμά μας, το εν λόγω username είναι το pfsense). Στη θυρίδα ακριβώς από κάτω πληκτρολογούμε και το αντίστοιχο password.



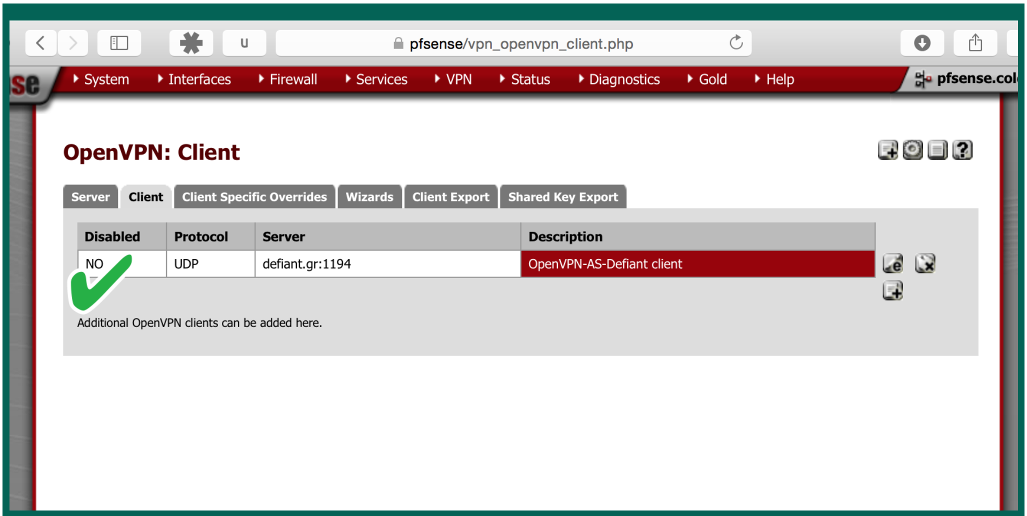
Εστιάζουμε τώρα την προσοχή μας στην ενότητα Cryptographic Settings. Στο περιβάλλον του pfSense φροντίζουμε ώστε να είναι τσεκαρισμένη η επιλογή Enable Authentication of TLS packets, αλλά όχι η Automatically generate a shared TLS authentication key. Από κάτω θα παρουσιαστεί ένα πλαίσιο, εντός του οποίου πρέπει να βάλουμε το κοινό κλειδί για την ταυτοποίηση των πακέτων TLS μεταξύ client και Access Server. Το κλειδί βρίσκεται μέσα στο αρχείο client.ovpn και συγκεκριμένα ανάμεσα στις ετικέτες <tls-auth> και </tls-auth>. Αντιγράφουμε το τμήμα που ξεκινά από το -----BEGIN openVPN Static key V1----- και φτάνει έως και το -----END openVPN Static key V1-----. Παραμένουμε στην ενότητα και για το Peer Certificate Authority επιλέγουμε το πιστοποιητικό του OpenVPN Access Server, το οποίο πριν λίγο εισαγάγαμε στο pfSense. Για το παράδειγμά μας, λοιπόν, ισχύει Peer Certificate Authority = OpenVPN-AS-CA-Defiant. Από κάτω, στην παράμετρο Client Certificate αναθέτουμε το πιστοποιητικό του client (κι αυτό το εισαγάγαμε στο pfSense, πριν λίγο). Για εμάς, ισχύει Client Certificate = OpenVPN-AS-Client. Φροντίζουμε εξάλλου ώστε Encryption algorithm = BF-CBC (128-bit) και Auth Digest Algorithm = SHA1 (160-bit).



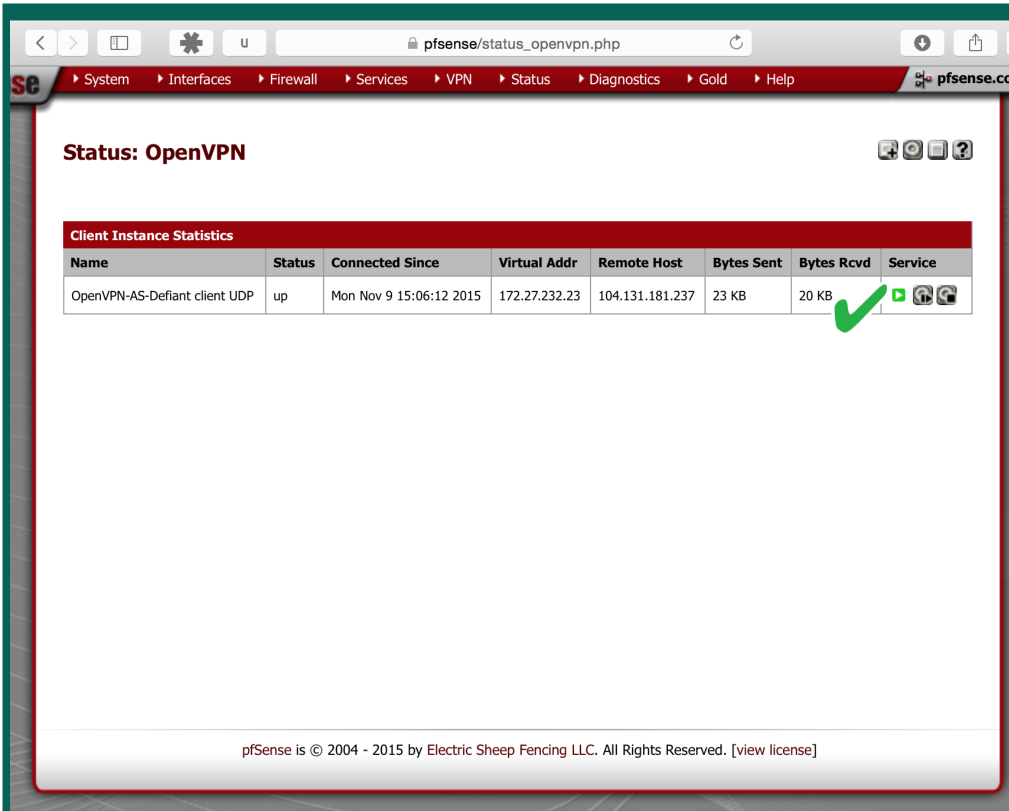
Την ενότητα Cryptographic Settings ακολουθεί η Tunnel Settings. Στο πλαίσιο IPv4 Tunnel Network περιγράφουμε το εικονικό δίκτυο (σε μορφή CIDR, **http://deltahacker.gr/?p=5138**) που υφίσταται μεταξύ client (ο router μας) και απομακρυσμένου OpenVPN Access Server. Πρόκειται για το δίκτυο 172.27.232.0/24. Πιο κάτω στην ίδια ενότητα θέτουμε Compression = Enabled with Adaptive Compression. Βεβαιωνόμαστε, τέλος, ότι η επιλογή Type-of-Service είναι τσεκαρισμένη.



Προκειμένου να επικυρώσουμε όλες τις ρυθμίσεις κι επιλογές για το παρόν client instance του OpenVPN, στο κάτω μέρος της σελίδας "OpenVPN: Client" πατάμε στο κουμπάκι Save.



Ιδού το client instance που μόλις ορίσαμε, με όνομα OpenVPN-AS-Defiant client. Στη στήλη Disabled, αριστερά, αναγράφεται NO — κι αυτό είναι καλό.

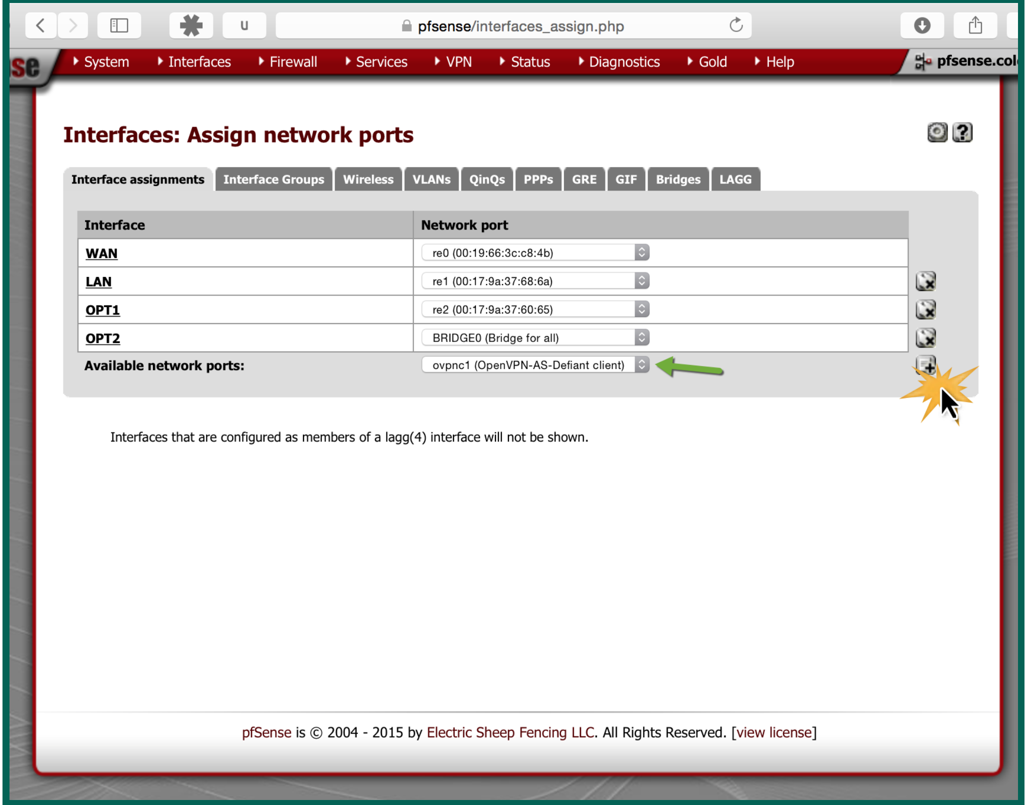


Στο περιβάλλον του pfSense δίνουμε Status → OpenVPN. Διαπιστώνουμε ότι ο router μας έχει συνδεθεί επιτυχώς στον απομακρυσμένο Access Server.

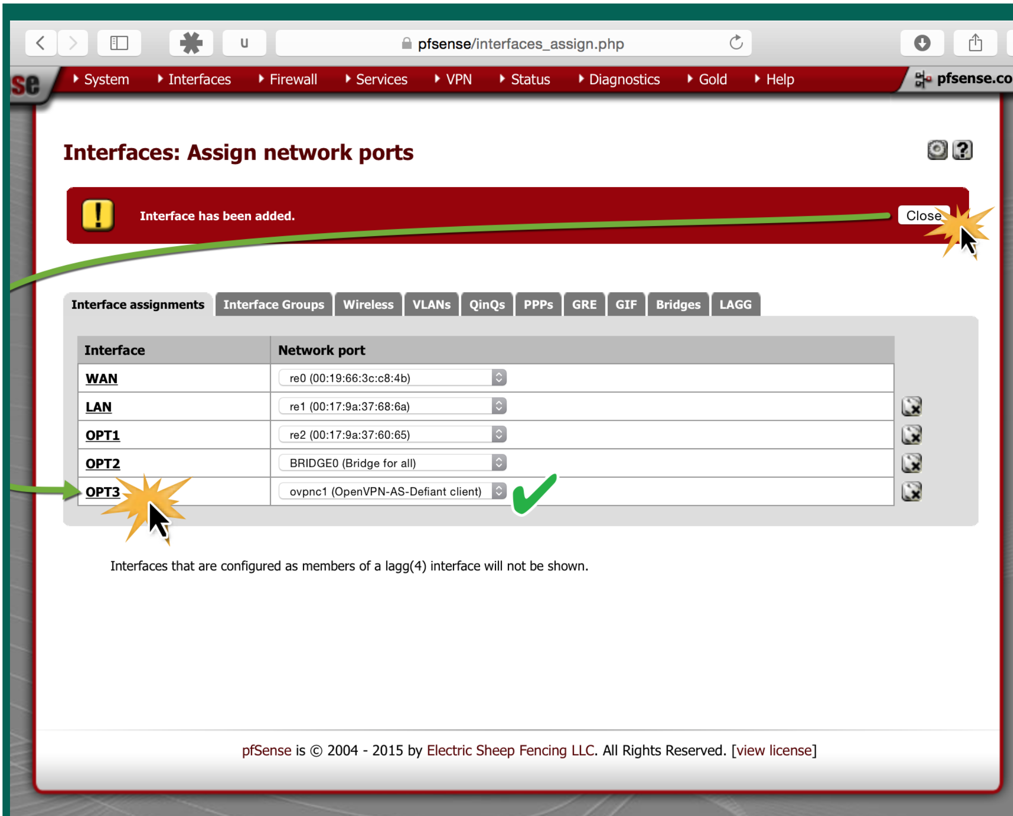
Το ότι ο router μας συνδέθηκε επιτυχώς στον OpenVPN Access Server είναι πολύ ευχάριστο, έχουμε όμως ένα μικρό θεματάκι ν' αντιμετωπίσουμε: Κανένα από τα μηχανήματα πίσω από τον router δεν μπορεί να βγει στο Internet. Είναι που ακόμη δεν έχουμε επέμβει στις ρυθμίσεις NAT του router. Συνεχίστε το διάβασμα.

Καθορισμός νέου interface για το VPN tunnel

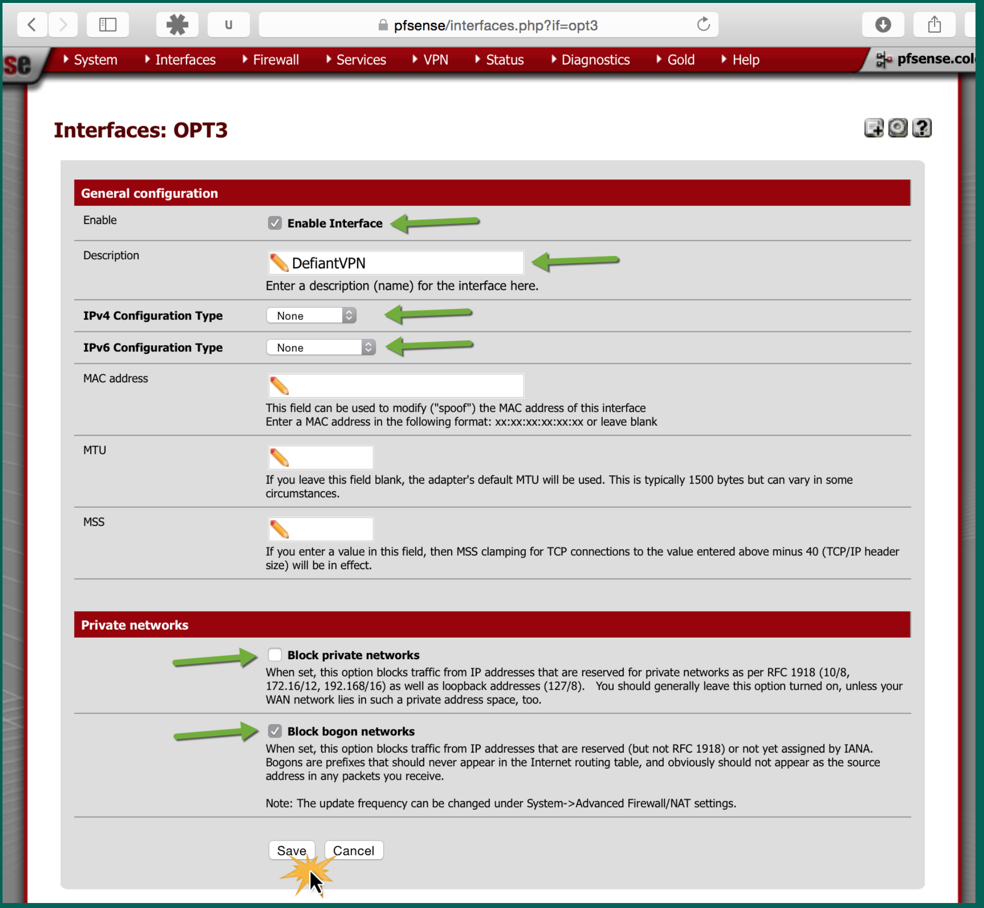
Πριν δούμε τις ρυθμίσεις NAT οφείλουμε να αντιστοιχίσουμε ένα όνομα στο νέο tunnelling network interface, μέσω του οποίου επιτυγχάνεται η κρυπτογραφημένη σύνδεση μεταξύ pfSense router και OpenVPN Access Server.



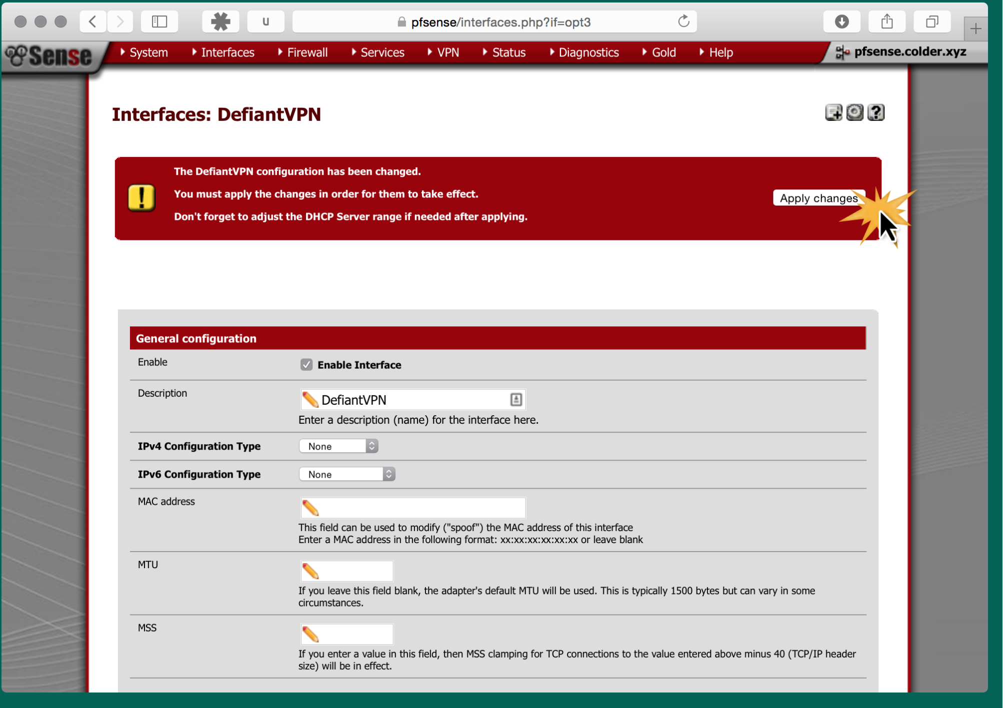
Στο περιβάλλον του pfSense επιλέγουμε Interfaces → (assign). Στη σελίδα που εμφανίζεται φαίνονται όλα τα ήδη ορισμένα network interfaces. Στο παράδειγμά μας βλέπουμε κατ' αρχάς το WAN, μέσω του οποίου ο router βγαίνει στο Internet. Υπάρχουν κι άλλα δύο, τοπικά interfaces, τα LAN και OPT1. Το δε OPT2 αποτελεί network bridge των LAN και OPT1 (βλ. **http://deltahacker.gr/?p=10500** και **http://deltahacker.gr/wp-admin/?p=10507**). Σε ένα απλούστερο setup το πιθανότερο είναι πως θα υπάρχουν δύο μόνο network interfaces: το WAN και το LAN. Σε κάθε περίπτωση, κάτω από τα ήδη ορισμένα network interfaces υπάρχει ένα μενού drop-down, στις επιλογές του οποίου συγκαταλέγεται και το interface που υφίσταται χάρη στο OpenVPN client instance. Στη δική μας περίπτωση, το προαναφερθέν interface είναι το ovpnc1 (OpenVPN-AS-Defiant client). Φροντίζουμε ώστε να 'ναι επιλεγμένο και κάνουμε ένα κλικ στο κουμπάκι με το "συν", κάτω δεξιά.



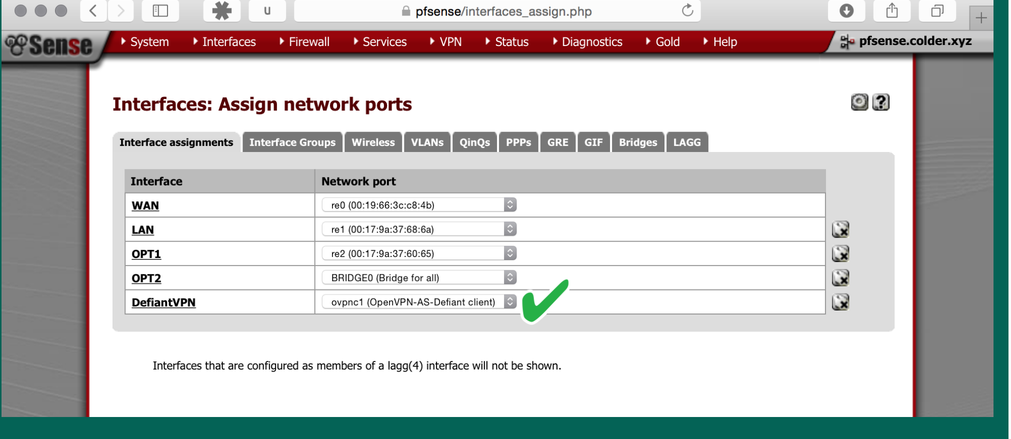
Το OPT3 είναι το όνομα που έδωσε το pfSense στο network interface που επιλέξαμε μόλις. Πατάμε στο κουμπάκι Close, στα δεξιά του πληροφοριακού μηνύματος στο πάνω μέρος της σελίδας, μετά πατάμε και στο OPT3.



Βρισκόμαστε τώρα στη σελίδα ρύθμισης του OPT3, του νέου network interface που μόλις προσθέσαμε στη σχετική λίστα του pfSense. Κατ' αρχάς, η επιλογή Enable Interface πρέπει να 'ναι τσεκαρισμένη. Στη θυρίδα από κάτω προτείνεται να πληκτρολογήσουμε ένα περιγραφικό όνομα για το interface. Σίγουρα κάποιοι δεν θα έχουν πρόβλημα με το OPT3, πιστεύουμε όμως ότι κάτι σαν το DefiantVPN σε βοηθά να θυμάσαι σε τι αφορά το interface, όσος χρόνος κι αν περάσει από την ημέρα που το όρισες. Φροντίζουμε εξάλλου ώστε να ισχύει IPv4 Configuration Type = None, όπως επίσης και IPv6 Configuration = None. Στην ενότητα Private networks, λίγο πιο κάτω, αμφότερα τα Block private networks και Block bogon networks πρέπει να 'ναι τσεκαρισμένα. Προσοχή: Αν το pfSense box δεν βγαίνει απευθείας στο Internet αλλά μπροστά του έχει ένα modem/router το οποίο αδυνατεί να λειτουργεί σε bridge mode, τότε το box βρίσκεται κι αυτό εντός private network και συνεπώς η επιλογή Block private networks προτείνεται να μην είναι τσεκαρισμένη. Διαφορετικά, η πρόσβαση στο web panel του modem/router δεν θα είναι δυνατή. Όλες τις αλλαγές και τις ρυθμίσεις που κάναμε τις επικυρώνουμε με ένα κλικ στο κουμπάκι Save.



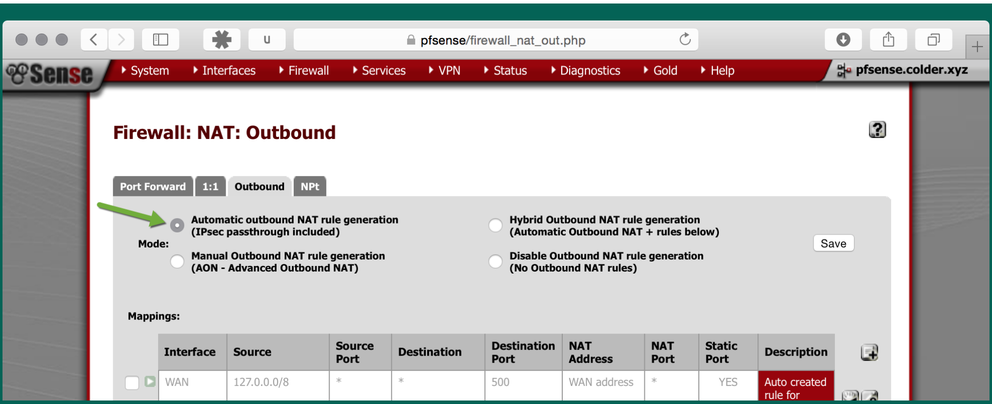
Το κλικ στο Save δεν είναι αρκετό για να παγιωθούν οι αλλαγές: Χρειάζεται κι άλλο ένα κλικ, στο κουμπί Apply changes.



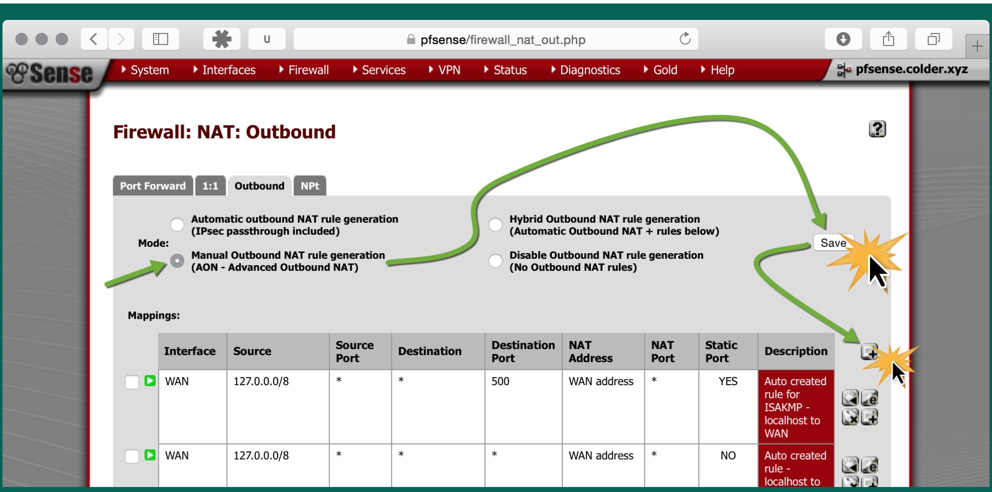
Το ολοκαίνουργιο network interface –στο παράδειγμά μας ονομάζεται DefiantVPN– έχει οριστεί κι ενεργοποιηθεί. Μέσω του συγκεκριμένου interface ο router μας επικοινωνεί με τον απομακρυσμένο OpenVPN Access Server, άρα κι όλα τα μηχανήματα πίσω του. Για την ακρίβεια, αυτό θα ισχύει αρκεί πρώτα να φροντίσουμε για τις ρυθμίσεις NAT.

Ρυθμίσεις outbound NAT

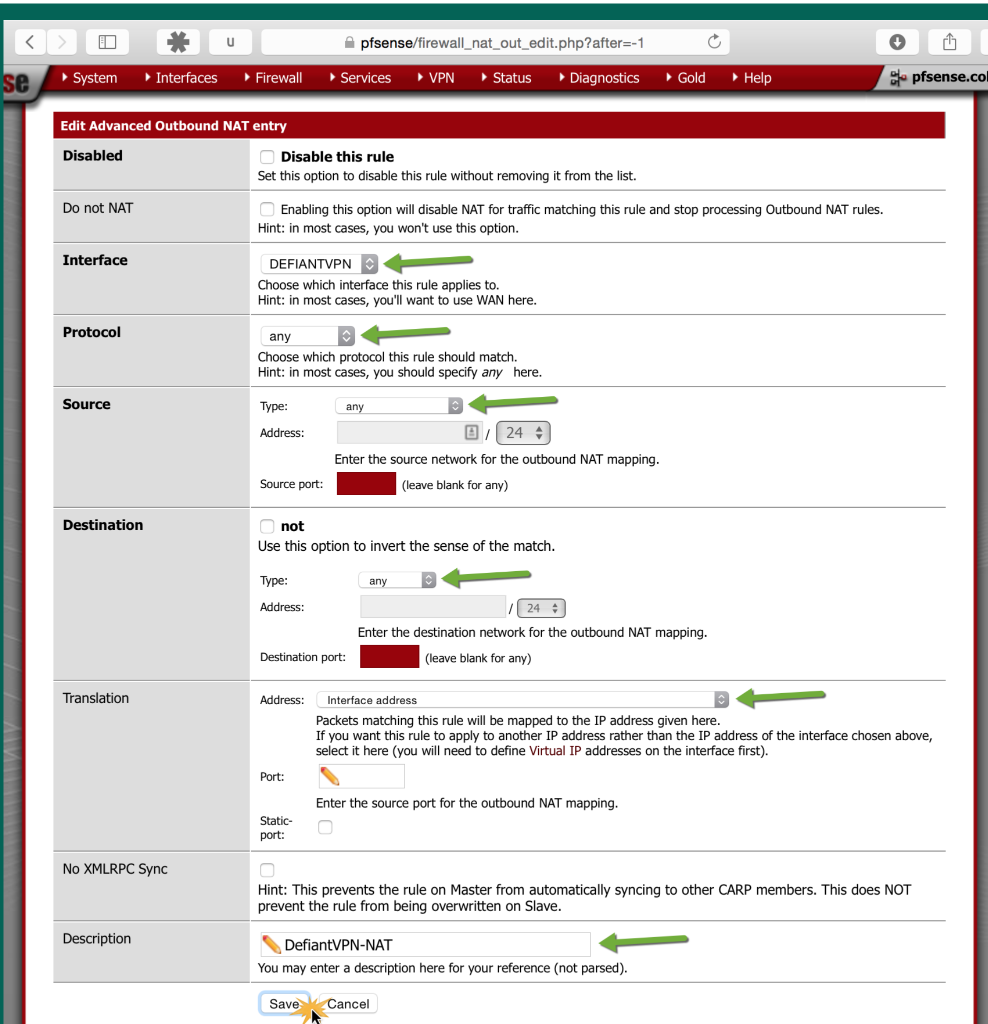
Το τι ακριβώς είναι το NAT και το γιατί πρέπει να επέμβουμε στις ρυθμίσεις του, είναι μια συζήτηση που την αφήνουμε για το σχετικό άρθρο στο επόμενο τεύχος του deltaHacker. Έχουμε πολλά ακόμη να πούμε και να κάνουμε για το tunnelling προς τον απομακρυσμένο OpenVPN Access Server, αλλά το παρόν άρθρο έχει ήδη μεγαλώσει επικίνδυνα. Προς το παρόν, αυτό που μας ενδιαφέρει άμεσα είναι να βγάλουμε όλους τους υπολογιστές και τις συσκευές του τοπικού δικτύου στο Internet, μέσω του απομακρυσμένου Access Server. Η συνέχεια στα screen shots που ακολουθούν.



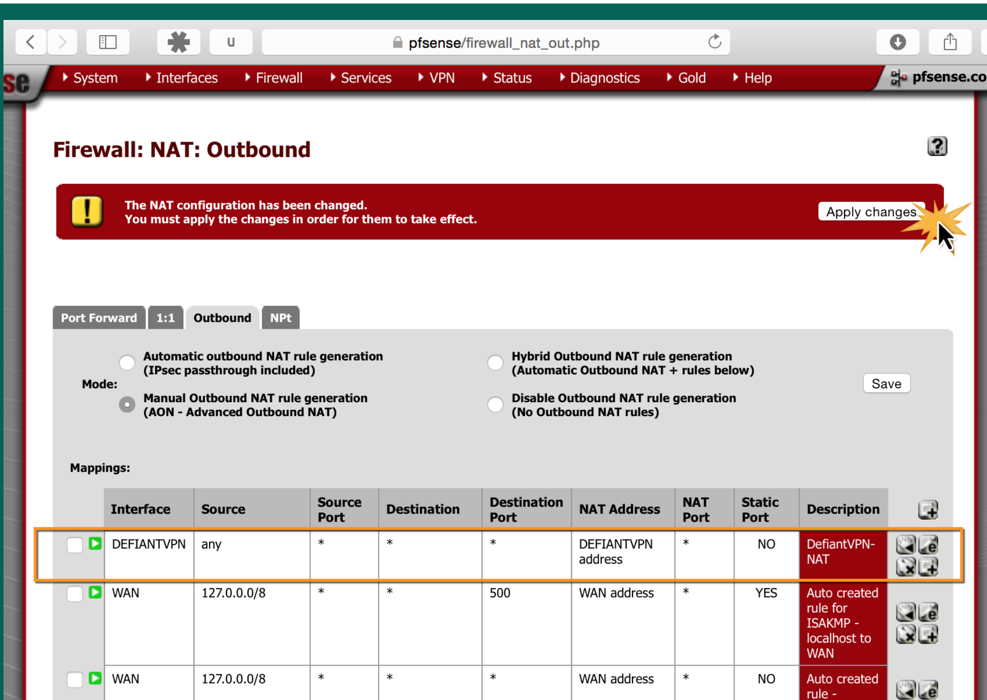
Από το περιβάλλον του pfSense ακολουθούμε τη διαδρομή Firewall → NAT → Outbound. Βλέπουμε ότι είναι επιλεγμένο το Automatic outbound NAT rule generation (IPsec passthrough included).



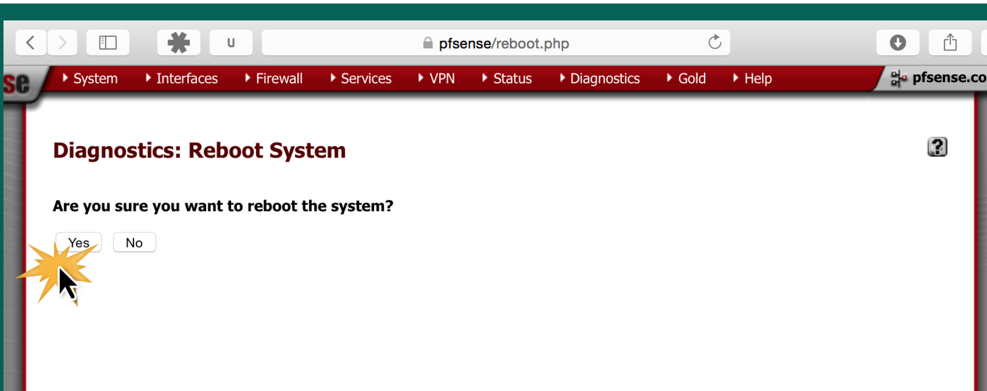
Επιλέγουμε το Manual Outbound NAT rule generation (AON - Advanced Outbound NAT) κι αμέσως πατάμε στο κουμπί Save, στα δεξιά της σελίδας. Ακολουθεί κι ένα κλικ στο κουμπάκι με το "συν", στα δεξιά της πρώτης (γκρι) γραμμής κάτω από το Mappings.



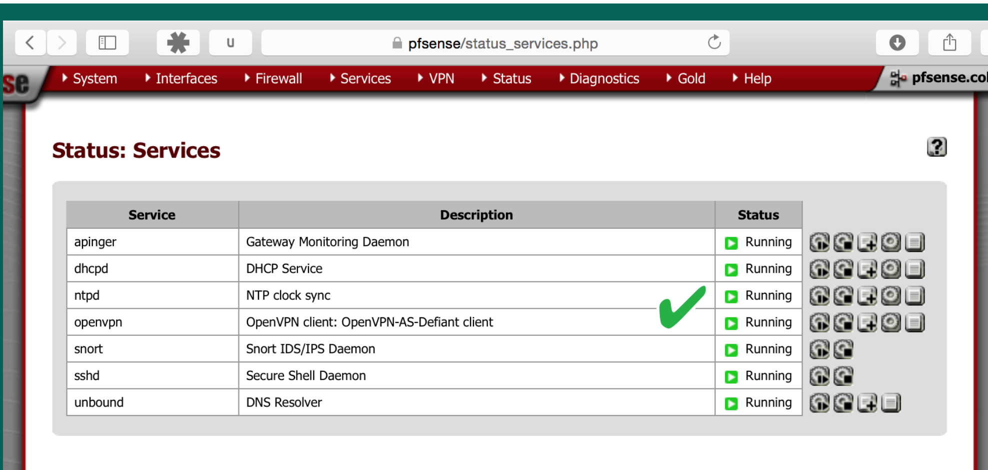
Στη νέα σελίδα που εμφανίζεται κάνουμε τις επιλογές που επιδεικνύονται από τα βελάκια. Μην ξεχάσετε να πληκτρολογήσετε κάτι χαρακτηριστικό στο πεδίο Description κι επικυρώστε με κλικ στο κουμπί Save.



Ο νέος κανόνας έχει προστεθεί και για να ληφθεί υπόψη αρκεί ένα κλικ στο κουμπί Apply Changes, πάνω δεξιά στη σελίδα.



Στο σημείο αυτό προτείνεται να επανεκκινήσουμε το pfSense box. Δίνουμε Diagnostics Reboot και στη σχετική ερώτηση απαντάμε καταφατικά, με κλικ στο Yes.



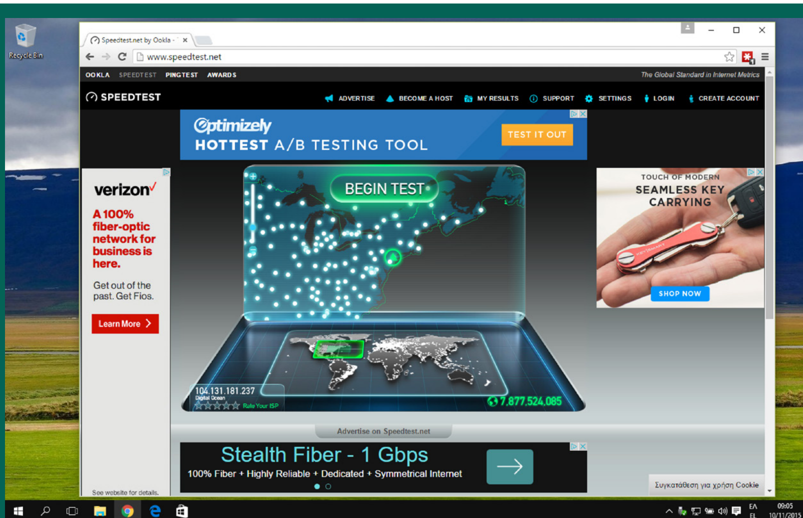
Μετά την εκκίνηση του pfSense box συνδεόμαστε στο web panel του κι επιλέγουμε Static → Services. Αν όλα έχουν πάει καλά, η υπηρεσία του OpenVPN θα πρέπει να 'ναι ενεργή και η σύνδεση στον απομακρυσμένο Access Server να υφίσταται.

`Έλεγχοι ορθής λειτουργίας

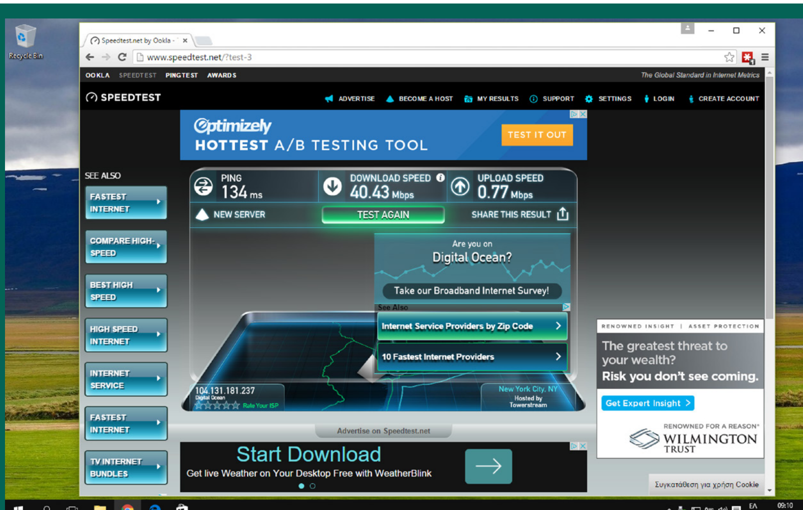
Εδώ που φτάσαμε, όλοι οι υπολογιστές και οι συσκευές πίσω από τον pfSense-based router βγαίνουν στο Internet από τον απομακρυσμένο OpenVPN Access Server, στέλνοντας και λαμβάνοντας δικτυακά πακέτα μέσω ενός κρυπτογραφημένου τούνελ το οποίο εκτείνεται μεταξύ του pfSense box και του Access Server. Ένας τρόπος για να διαπιστώσουμε πως όλα έχουν πάει καλά, είναι να επισκεφθούμε ένα site σαν το https://www.whatismyip.com. Αξίζει επίσης να κάνουμε και μια βόλτα από το http://www.speedtest.net.

Η συνέχεια

Καλορίζικο το νέο, αναβαθμισμένης ιδιωτικότητας, setup για το LAN! Υπάρχουν πολλά που αξίζει να δούμε ακόμα, αν μη τι άλλο για να καταλάβουμε καλύτερα τι γίνεται παρασκηνιακά, αλλά και για να έχουμε καλύτερο έλεγχο πάνω στα δύο κανάλια επικοινωνίας που υφίστανται, πλέον, προς το Internet: το ένα είναι κρυπτογραφημένο και καταλήγει στον OpenVPN Access Server μας, το άλλο δεν είναι κρυπτογραφημένο και καταλήγει στον ISP μας. Σε άρθρο του επόμενου τεύχους ασχολούμαστε με τα ακόλουθα ζητήματα: outbound NAT, HMAC firewall, aliases, επιλεκτική δρομολόγηση μέσω OpenVPN tunnel αναλόγως συσκευής ή/και site, mtr tests, packet capturing κ.ά. Έως τότε, καλή σας συνέχεια – και καλή σας διασκέδαση.



Από έναν οποιονδήποτε υπολογιστή πίσω από τον pfSense-based router, επισκεφτόμαστε το **http://www.speedtest.net**. Τα πάντα –ο χάρτης, το αναγραφόμενο δημόσιο IP, ακόμη και οι διαφημίσεις–, φανερώνουν ότι βγαίνουμε στο Internet μέσω του OpenVPN Access Server ο οποίος, παρεμπιπτόντως, βρίσκεται σε data center κάπου στη Νέα Υόρκη.



Τα αποτελέσματα ενός speed test είναι παράξενα: ενώ η ταχύτητα download είναι άριστη, εκείνη του upload είναι μάλλον τραγική. Γιατί συμβαίνει αυτό; Στην πραγματικότητα, η ταχύτητα upload που έχουμε μέσω του ISP μας είναι πολύ κοντά σ' αυτή που μέτρησε το Speedtest. Η δε πραγματική ταχύτητα download είναι περί τα 12,5Mbps, όμως για κάποιο λόγο το Speedtest μέτρησε την ταχύτητα μεταξύ του OpenVPN Access Server, ο οποίος είναι στη Νέα Υόρκη, κι ενός άλλου server, επίσης στη Νέα Υόρκη.


COOKIES
ΥΠ.ΠΡΟ.ΠΟ, ΟΚΤΩΒΡΗΣ 2015
ΠΡΕΠΕΙ ΝΑ ΚΑΤΑΦΕΡΟΥΜΕ ΕΝΑ ΣΥΝΤΡΙΠΤΙΚΟ ΧΤΥΠΗΜΑ ΣΤΗ ΔΙΑΧΥΤΗ ΑΝΟΜΙΑ, ΤΗ ΔΙΑΦΘΟΡΑ ΚΑΙ ΤΗ ΔΙΑΠΛΟΚΗ! ΑΚΟΥΩ ΙΔΕΕΣ!
ΠάNOS
ΥΠΑΡΧΕΙ ΕΚΕΙΝΟΣ Ο ΛΑΘΡΕΜΠΟΡΟΣ ΠΟΥ...
ΓΑΜΑ ΤΟ, ΔΕΝ ΠΑΙΖΕΙ ΑΥΤΟ!
ΑΝ ΑΝΟΙΓΑΜΕ ΤΟΝ ΦΑΚΕΛΟ ΜΕ ΤΑ ΔΑΝΕΙΑ ΤΩΝ ΜΜΕ;
ΑΥΤΟ ΚΑΛΥΤΕΡΑ ΓΑΜΑ ΤΟ...
ΕΙΝΑΙ ΚΙ ΕΝΑΣ ΕΦΟΠΛΙΣΤΗΣ Ο ΟΠΟΙΟΣ...
ΩΠΑ ΩΠΑ, ΓΑΜΑ ΤΟ, ΜΗΝ ΤΟ ΠΕΙΣ ΚΑΝ!
...ΧΜΜΜ...
gamotu
ΜΑΓΚΕΣ ΤΟ'ΧΩ!

deltahacker.gr